Michael Leibert
Math 504
Homework 11

2. Let

$$F(x) = \int_0^x dz \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-z^2/2}. \tag{1}$$

$F(x)$ is an important function - essentially the cdf of a normal random variable - that has no analytic formula and must be evaluated numerically. Write a function, **Fapprox**$(n, method)$ that approximates $F(\infty)$ by numerical integration. If **method** is set to the character string "reimann" or "trapezoid" use a Riemann sum and trapezoid rule method, respectively, with n grid points. If **method** is set to "useR", use the R integrate function with subdivisions set to n. (R's integrate function uses an adaptive grid method. In these methods, the grid is made dense in regions where the function varies, see Sauer for further details on such methods.)

Since you cannot integrate to ∞, you must pick some reasonable cutoff. We know that the true value is $F(\infty) = 1/2$. Consider approximating the integral using each of the three methods, try $n = 10, 100, 1000, 10000$ and compare accuracy.

```
#Dropped the error in the R's integrate function to store value in dataframe
Fapprox<-function(n,method){
        Fx<-function(x){(1/sqrt(2*pi)) *  exp((-x^2) / 2) }
        if (method == "useR") {return(integrate(Fx,0,Inf,subdivisions=n)$value) #£
        } else if (method == "reimann")  {
                h=6/n;a=0;i=0:(n-1); return( sum( Fx(a+i*h)*h ))
        } else if (method == "trapezoid")       {
                h=6/n;a=0;i=0:(n-1); return( sum( ( Fx(a+i*h) +
                 Fx(a+(i+1)*h) ) / 2 * h ))
        } else (return("Give me useR, reimann, or trapezoid please."))}

fapprox<-data.frame(rep(NA,3))
method<-c("reimann","trapezoid","useR")

for( i in 1:4){ fapprox[,i]<-c( Fapprox(10^i,method[1]) ,
    Fapprox(10^i,method[2]), Fapprox(10^i,method[3]) ) }

rownames(fapprox)<-method;colnames(fapprox)<-paste0("10^",1:4)
fapprox
```

```
##                10^1      10^2      10^3      10^4
## reimann   0.6196827 0.5119683 0.5011968 0.5001197
## trapezoid 0.5000000 0.5000000 0.5000000 0.5000000
## useR      0.5000000 0.5000000 0.5000000 0.5000000
```

So clearly as we increase the n for the Reimann and trapezoid methods, the numerical approximation becomes more accurate. What is interesting however, because most of the area for $F(x)$ (as specified by our integral) is between x=0 and around x=3.5, the greater the choice of the upper limit has a significant effect on how accurate the Reimann and trapezoid methods are. For a cutoff around x=6, it gives nice results at all n levels. If we push the cutoff to x=100, at $n = 10$ the estimates for Reimann and Trapezoid balloon from 0.62 and .5 to 4 and 2, respectively. The rectangles are picking up a lot of error past around x=6. As we increase the n, it becomes less of a problem because the rectangles end up being products of very small numbers, and the trapezoid method ends up converging to 0.5 (Reimann around 0.5001197). Likewise, if we set the cutoff too low, say around x=3 or x=4, we miss some of the area after x=4. So the estimates never quite reach 0.5.

3. This problem repeats the spline regression problems of hw 11, but now we will use many knots. Specifically, consider the case 1000 knots equally spaced between the minimum and maximum x values (ages) of the dataset. As before, for this problem let's consider solely the female portion of the BoneMass dataset (file attached).

   To manipulate B-splines, R provides the function **splineDesign** as part of the **splines** package. Here are the two ways you will need to call **splineDesign** to apply spline regression with a penalty method.

```
B <- splineDesign(knots=myknots, x=x, outer.ok=T)
Bpp <- splineDesign(knots=myknots, x=mygrid, derivs=2,
outer.ok = T)
```

   Above, B will be a design matrix, meaning that $B_{ij} = b_j(x_i)$ where $b_j(x)$ is the jth spline function in the B-spline basis and x_i is the ith x sample from the dataset. Bpp is similar, except that given the flag **derivs=2**, $Bpp_{ij} = b_j''(x_i)$ (i.e. the second derivative of $b_j(x)$). See below for the meaning of **mygrid**. Note: Using **splineDesign**, the dimension of the spline space is $K-4$, where K is the number of knots, rather than $K+4$ as when we used **bs** in hw 11. **splineDesign** places some constraints on the behavior of the splines at the end points that restrict the dimension to $K-4$ rather than $K+4$. This issue is minor and does not change the computations that must be done.. B will have dimension N by $K-4$ where N is the dimension of x. Bpp will have $K-4$ columns, but the number of rows will depend on the number of values in mygrid.

   (a) Consider minimizing the following penalized least squares, which we discussed in class

$$\min_{\alpha} \sum_{i=1}^{N} \|y - \sum_{j=1}^{K-4} \alpha_j b_j(x_i)\|^2 + \rho \int_{x_{\min}}^{x_{\max}} \left(\left(\sum_{j=1}^{K-4} \alpha_j b_j(z) \right)'' \right)^2 dz \tag{2}$$

   Show that this reduces to calculating

$$\min_{\alpha} \|y - B\alpha\|^2 + \rho \alpha^T \Omega \alpha \tag{3}$$

   where B is precisely the matrix returned by **splineDesign** above and where Ω is a $K-4$ by $K-4$ matrix given by

$$\Omega_{k\ell} = \int_{x_{\min}}^{x_{\max}} b_k''(z) b_\ell''(z) dz \tag{4}$$

   Show that the solution to the minimization is given by

$$\alpha = (B^T B + \rho\Omega)^{-1} B^T y \tag{5}$$

$$\min_{\alpha} \underbrace{\sum_{i=1}^{N} \|y - \sum_{j=1}^{K-4} \alpha_j b_j(x_i)\|^2}_{\text{Fit Term}} + \underbrace{\rho \int_{x_{\min}}^{x_{\max}} \left(\left(\sum_{j=1}^{K-4} \alpha_j b_j(z) \right)'' \right)^2 dz}_{\text{curviness term}}$$

   For our fit term we vectorize. Let y be the vector of the responses, y_i, and let B be the model matrix from **splineDesign**, where $B_{ij} = h_j(x_i)$ and we know this term can be written as

$$\sum_{i=1}^{N} \|y - \sum_{j=1}^{K-4} \alpha_j b_j(x_i)\|^2 = ||y - B\alpha||^2.$$

$$
\begin{aligned}
&\min_{\alpha} ||y - B\alpha||^2 + \rho \int_{x_{\min}}^{x_{\max}} \left(\left(\sum_{j=1}^{K-4} \alpha_j b_j(z) \right)'' \right)^2 \, \mathrm{d}z \\
&= \min_{\alpha} ||y - B\alpha||^2 + \rho \int_{x_{\min}}^{x_{\max}} \left(\sum_{j=1}^{K-4} \alpha_j b_j''(z) \right)^2 \, \mathrm{d}z \\
&= \min_{\alpha} ||y - B\alpha||^2 + \rho \int_{x_{\min}}^{x_{\max}} \left(\sum_{j=1}^{K-4} \alpha_j b_j''(z) \cdot \sum_{j=1}^{K-4} \alpha_j b_j''(z) \right) \, \mathrm{d}z \\
&= \min_{\alpha} ||y - B\alpha||^2 + \rho \int_{x_{\min}}^{x_{\max}} \sum_{j=1}^{K-4} \sum_{\ell=1}^{K-4} \alpha_j \alpha_\ell b_j''(z) b_\ell''(z) \, \mathrm{d}z \\
&= \min_{\alpha} ||y - B\alpha||^2 + \rho \sum_{j=1}^{K-4} \sum_{\ell=1}^{K-4} \alpha_j \alpha_\ell \underbrace{\left(\int_{x_{\min}}^{x_{\max}} b_j''(z) b_\ell''(z) \, \mathrm{d}z \right)}_{\substack{\Omega_{j\ell} \text{ where } \Omega \text{ is a} \\ (K-4) \times (K-4) \text{ matrix.} \\ \text{equation (4)}}} \\
&= \min_{\alpha} ||y - B\alpha||^2 + \rho \sum_{j=1}^{K-4} \sum_{\ell=1}^{K-4} \alpha_j \alpha_\ell \, \Omega_{j\ell} \\
&= \min_{\alpha} ||y - B\alpha||^2 + \rho \left(\alpha^T \Omega \alpha \right) \qquad \text{equation (3)} \\
&= \min_{\alpha} (y - B\alpha) \cdot (y - B\alpha) + \rho \alpha^T \Omega \alpha \\
&= \min_{\alpha} \; y \cdot y - 2 \left(B^T y \right)^T \alpha + \alpha^T B^T B \alpha + \rho \alpha^T \Omega \alpha \\
&= \min_{\alpha} \; y \cdot y - 2 \left(B^T y \right)^T \alpha + \alpha^T \left(B^T B + \rho \, \Omega \right) \alpha
\end{aligned}
$$

Where we can solve for $\alpha = \left(B^T B + \rho \, \Omega \right)^{-1} B^T y$; equation (5).

(b) Use R to calculate Ω through numerical integration. The matrix *Bpp* gives values of $b_j''(z)$ on a grid of z values. The vector **mygrid** (in code for *Bpp* above) gives the z values at which b_j'' is calculated. For simplicity, calculate Ω using Riemann integration (see below). Make sure **mygrid** forms a dense grid of values so that you get accurate estimates using Riemann integration.

```
require(splines)
## Loading required package:  splines
bones<-read.table("BoneMassData.txt",header=T)
bones<-bones[which(bones$gender == "female"),]
n=1000

k<-seq(min(bones$age),max(bones$age),length.out = n)
B <- splineDesign(knots= k,  x=bones$age, outer.ok=T)
z<-seq(min(bones$age),max(bones$age),   length.out = 10000 )
Bpp <- splineDesign(knots=k, x=z, derivs=2, outer.ok = T)

#calculate Omega through numerical integration
h<-z[2]-z[1]
omega<-h*(t(Bpp) %*% Bpp )
```

(c) Show that the matrix B^TB is not invertible but that $B^TB + \rho\Omega$ is for $\rho > 0$. You can do this by just computing the determinant for various values of ρ in R or through theoretical arguments.

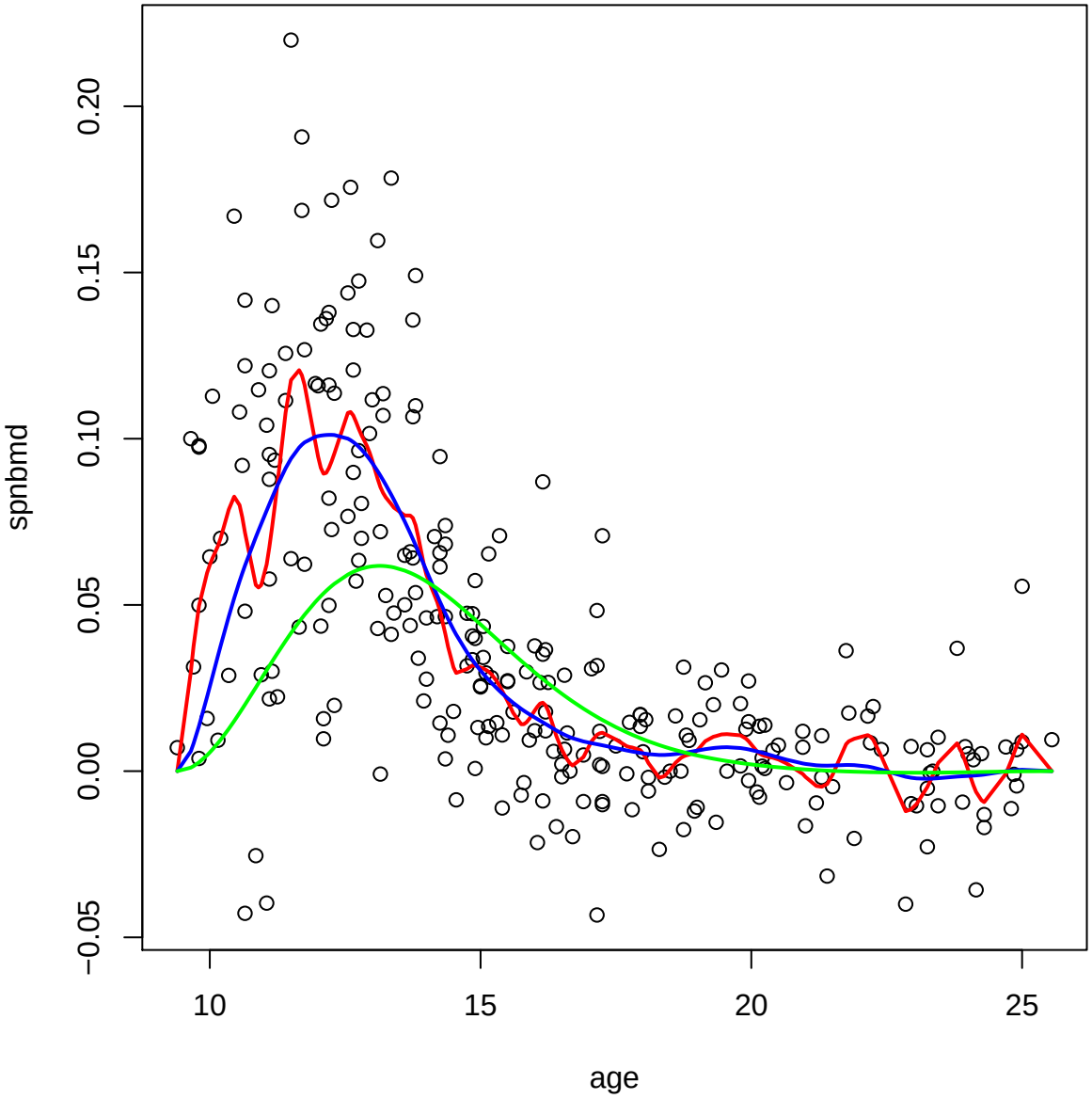
Note the determinant is equal to `Inf` because it is too large to be stored in floating point. So the matrix $B^TB + \rho\Omega$, $\rho > 0$, is invertible.

```
h<-z[2]-z[1]
omega<-h*(t(Bpp) %*% Bpp )

#invertible?
det(solve(t(B)%*%B))                    #NO
## Error in solve.default(t(B) %*% B): Lapack routine dgesv:  system is exactly singular:
U[1,1] = 0
det( ( t(B)%*%B + (.01 * omega)))       #YES
## [1] Inf
det( ( t(B)%*%B + (1 * omega))) #YES
## [1] Inf
det( ( t(B)%*%B + (10 * omega)))        #YES
## [1] Inf
det( ( t(B)%*%B + (1001 * omega)))      #YES
## [1] Inf
```

(d) Calculate α for (a) $\rho = .01$, (b) $\rho = 1$ and (c) $\rho = 100$. In each case plot the smoothing spline (hint: B provides you discretized versions of the $b_j(x)$, linearly combine them using α to form the fitted spline) and the data points. Comment on the fit in each case.

```
h<-z[2]-z[1];omega<-h*(t(Bpp) %*% Bpp )

y<-as.matrix( bones[,4] )
A<-matrix(NA,(n-4),3)
A[,1]<-solve( t(B)%*%B + (.01 * omega) ) %*% t(B)%*% y
A[,2]<-solve( t(B)%*%B + (1 * omega) ) %*% t(B)%*% y
A[,3]<-solve( t(B)%*%B + (100 * omega) ) %*% t(B)%*% y
```

```
par(mar=c(4.1,4.1,1,1))
datt<-data.frame(bones$age, B%*% A[,1], B%*% A[,2], B%*% A[,3] )
datt<-datt[with(datt, order(bones$age)), ]

plot(bones[,2], bones[,4], xlab="age", ylab="spnbmd")
lines( datt[,1],datt[,2],col="red", lwd=2)
lines( datt[,1],datt[,3],col="blue", lwd=2)
lines( datt[,1],datt[,4],col="green", lwd=2)
```



For the red line, ρ is small, and we see what could be considered over-fit. This function has a lot of curviness. For the blue like, ρ is somewhat in the middle with regards to the penalty. It shows some curviness, but not nearly as much as the red line, yet it has more fit than the green line. For the green line, ρ is somewhat large, implementing a large penalty for the function and reducing the fit.

4. Let the function space $\mathcal{F}$ be composed of all polynomials of degree 6 or less. Consider the dataset composed of the 11 points (x_i, y_i) where $y_i = \sin(x_i)$ and $x_i = i - 1$ for $i = 1, 2, \ldots, 11$.

   (a) Find a basis $h_1(x), h_2(x), \ldots, h_D(x)$ for $\mathcal{F}$. What is D?

   A basis for $\mathcal{F}$ is $h_1(x) = 1,\ h_2(x) = x,\ h_3(x) = x^2,\ h_4(x) = x^3,\ h_5(x) = x^4,\ h_6(x) = x^5,\ h_7(x) = x^6$, where $D = 7 = \dim(\mathcal{F})$.

   (b) Consider the least squares regression

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{11} (y_i - f(x_i))^2 \tag{6}$$

   We can express $f(x)$ in the $h_i(x)$ basis, $f(x) = \sum_{i=1}^{D} \alpha_i h_i(x)$. Using this expression for $f(x)$, we have shown that the optimal $f(x)$ can be found by solving for α:

$$\min_{\alpha \in \mathbb{R}^D} \|y - B\alpha\|, \tag{7}$$

   where B is the model matrix. What are the dimensions of B? Solve for α and plot the fitted $f(x)$ and the data.

$$\dim(B) = 11 \times 7$$

$$B = \begin{pmatrix} b_1(x_1) & b_2(x_1) & \ldots & b_7(x_1) \\ b_1(x_2) & b_2(x_1) & \ldots & b_7(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_{11}) & b_2(x_{11}) & \ldots & b_7(x_{11}) \end{pmatrix}$$

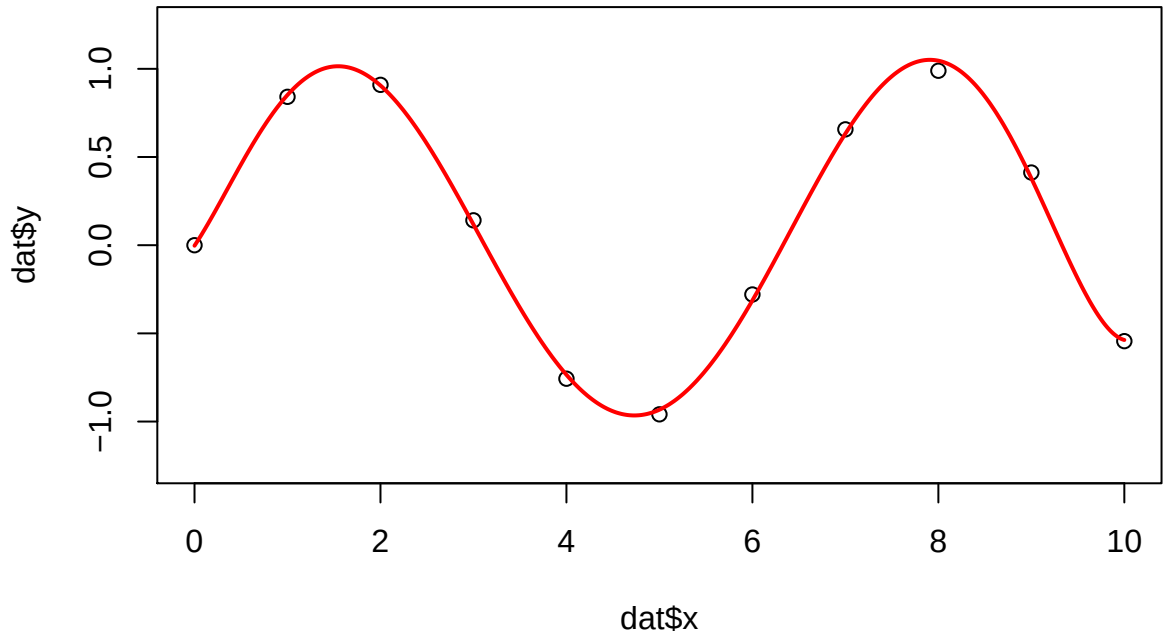
$$11 \times 7$$

```
dat<-data.frame(sin(0:10),0:10)
names(dat)<-c("y","x")

model_matrix <- function(x) {
        nx <- length(x)
        m <- cbind(rep(1, nx), x, x^2, x^3, x^4, x^5, x^6 )
        colnames(m )<-NULL
        return(m)       }

B<-  model_matrix(dat$x)
alpha <- solve(t(B) %*% B, t(B) %*% as.matrix(dat$y));alpha
```

```
##                [,1]
## [1,] -0.0028844502
## [2,]  0.7953252487
## [3,]  0.5111799590
## [4,] -0.5905649976
## [5,]  0.1535989224
## [6,] -0.0154630321
## [7,]  0.0005412731
```

```
x_grid <- seq(min(dat$x), max(dat$x), .01)
B_grid <- model_matrix(x_grid)
y_grid <- B_grid %*% alpha
```

```
par(mar=c(4.1,4.1,1,1))
plot(dat$x, dat$y , ylim=c(-1.25,1.25) )
lines(x_grid, y_grid, col="red", lwd=2) #£
```



(c) Repeat (b) but now consider only the first D datapoints. What are the dimensions of B? Solve for $f(x)$ and plot the fitted $f(x)$ and the data.

$$\dim(B) = 7 \times 7$$

$$B = \begin{pmatrix} b_1(x_1) & b_2(x_1) & \dots & b_7(x_1) \\ b_1(x_2) & b_2(x_1) & \dots & b_7(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_7) & b_2(x_7) & \dots & b_7(x_7) \end{pmatrix}$$

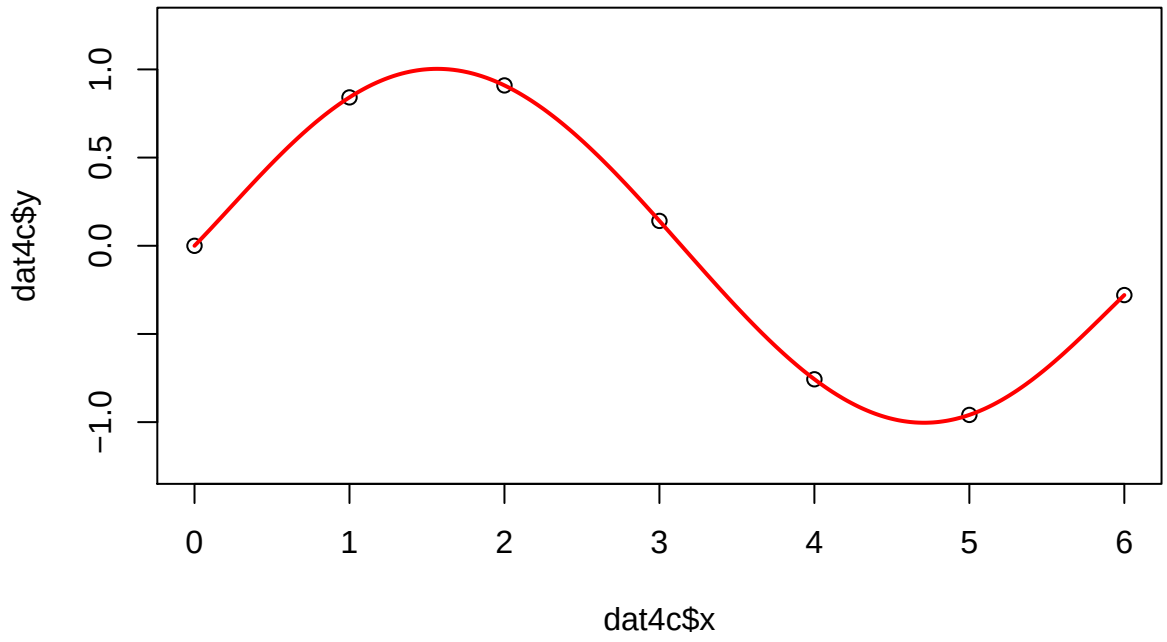
$$7 \times 7$$

```
D=7
dat4c<-dat[1:D,]
B4c<-  model_matrix(dat4c$x)
alpha4c <- solve( B4c,  as.matrix(dat4c$y));alpha4c
```

```
##                [,1]
## [1,]  0.0000000000
## [2,]  0.9037647389
## [3,]  0.2254590209
## [4,] -0.3576634860
## [5,]  0.0731892930
## [6,] -0.0031262599
## [7,] -0.0001523221
```

```
x_grid <- seq(min(dat4c$x), max(dat4c$x), .01)
B_grid <- model_matrix(x_grid)
y_grid <- B_grid %*% alpha4c
```

```
par(mar=c(4.1,4.1,1,1))
plot(dat4c$x, dat4c$y , ylim= c(-1.25,1.25))
lines(x_grid, y_grid, col="red", lwd=2) #£
```



(d) Repeat (b) but now consider only $D-2$ datapoints. What is the dimension of B? Find two α that satisfy $B\alpha = y$ and plot both resultant $f(x)$ and the data. (Hint: To find an α, assume that some of the coordinates of α are zero, so that you are implicitly ignoring some columns of B.)

$$\dim(B) = 5 \times 7$$

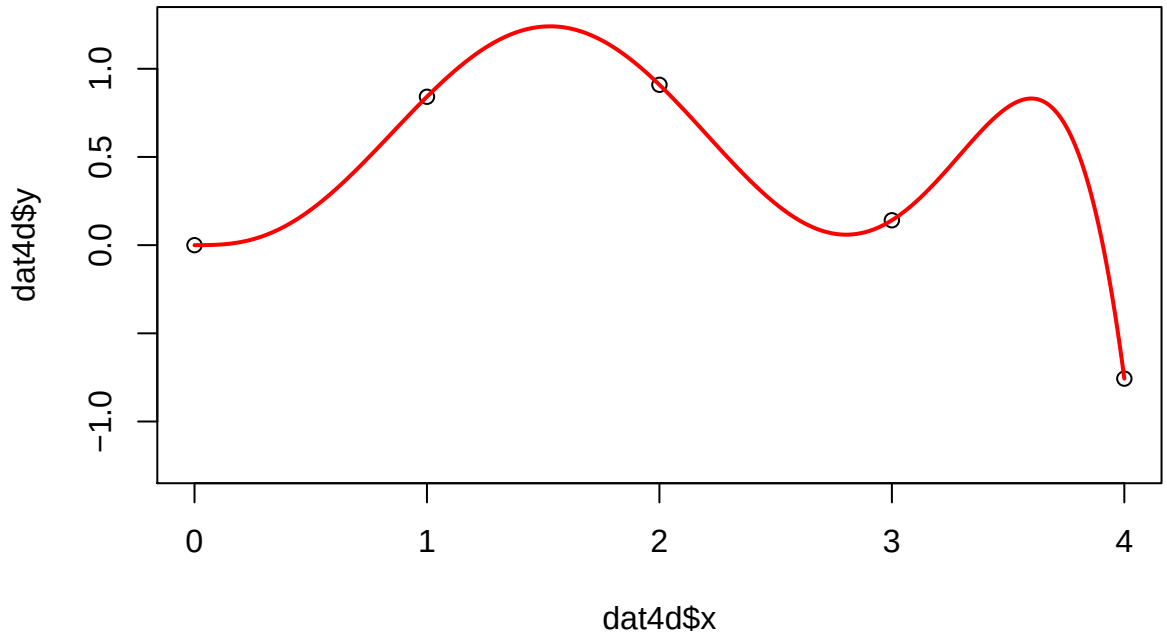
$$B = \begin{pmatrix} b_1(x_1) & b_2(x_1) & \dots & b_7(x_1) \\ b_1(x_2) & b_2(x_1) & \dots & b_7(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_5) & b_2(x_5) & \dots & b_7(x_5) \end{pmatrix}$$

$$5 \times 7$$

```
dat4d<-dat[1:(D-2),]
B4d<-  model_matrix(dat4d$x)
a<-c(2,3)
B4d<-  B4d[,-a]

alpha4d <- solve(t(B4d) %*% B4d, t(B4d) %*% as.matrix(dat4d$y) );alpha4d

##                [,1]
## [1,] -6.939686e-11
## [2,]  2.716643e+00
## [3,] -2.624849e+00
## [4,]  8.376761e-01
## [5,] -8.799825e-02

x_grid <- seq(min(dat4d$x), max(dat4d$x), .01)
```

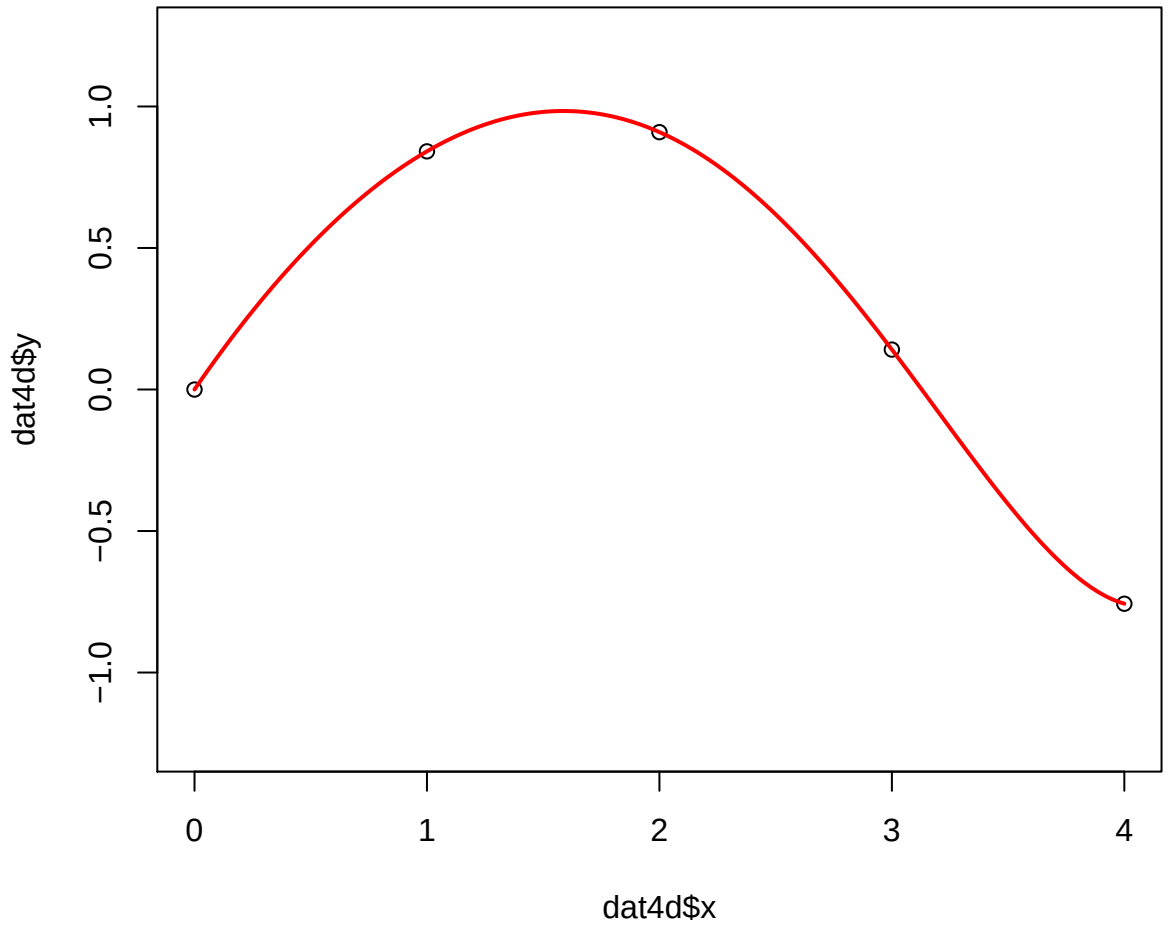
```
B_grid <- model_matrix(x_grid)
y_grid <-B_grid[,-a] %*% alpha4d
```

```
par(mar=c(4.1,4.1,1,1))
plot(dat4d$x, dat4d$y , ylim= c(-1.25,1.25))
lines(x_grid, y_grid, col="red", lwd=2)     #£
```




```
dat4d<-dat[1:(D-2),]
B4d<-  model_matrix(dat4d$x)
a<-sample( 1:7,2)
a<-c(4,5)
B4d<-  B4d[,-a]

alpha4d <- solve(t(B4d) %*% B4d, t(B4d) %*% as.matrix(dat4d$y) );alpha4d

##               [,1]
## [1,]  1.115944e-14
## [2,]  1.197020e+00
## [3,] -3.518894e-01
## [4,] -4.907417e-03
## [5,]  1.247691e-03

x_grid <- seq(min(dat4d$x), max(dat4d$x), .01)
B_grid <- model_matrix(x_grid)
y_grid <-B_grid[,-a] %*% alpha4d
```

```
par(mar=c(4.1,4.1,1,1))
plot(dat4d$x, dat4d$y , ylim= c(-1.25,1.25))
lines(x_grid, y_grid, col="red", lwd=2)#£
```



(e) Compare the solutions $f(x)$ in (b)-(d) and discuss how the solutions correspond to the relationship between the number of datapoints and the dimension of $\mathcal{F}$.

For the solution $f(x)$ in 4b the B matrix is a thin matrix, that is $D < N$. For matrices of this type, $B\alpha = y$ has no solution in general. So in order to find an $f(x)$ we find an α that minimizes $||B\alpha - y||^2$, which is solved by the normal equations $\alpha = \left(B^T B\right)^{-1} B^T y$. In a sense we are doing regression; $f(x)$ will not be equal to y for every value. For the solution $f(x)$ in 4c the B matrix is a square matrix, that is $D = N$. Here, $B\alpha = y$ has a unique solution, $\alpha = B^{-1}$. In this case $f(x)$ is hitting every single point. In the last case we have a fat matrix, that is $D > N$. There are infinite many solutions in this case, and we showed two of those solutions. This is a case that is high dimensional statistics.