```
#> Warning: package 'ggplot2' was built under R version 3.5.3
#> Warning: package 'tidyr' was built under R version 3.5.3
```

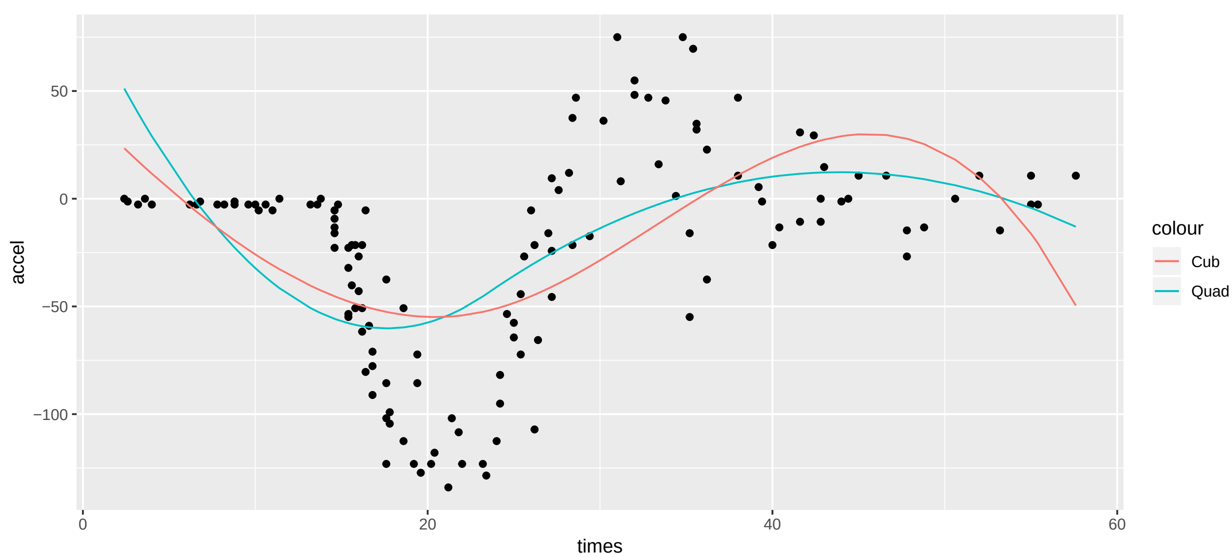
Praktikum 8

Aufgabe 1

a)

```r
quad = lm(accel ~ times + I(times^2) + I((times-median(times))^2*(times>median(times))), data=mcycle)
cub = lm(accel ~ times + I(times^2)+ I(times^3) + I((times-median(times))^3*(times>median(times))), data
summary(quad)
#>
#> Call:
#> lm(formula = accel ~ times + I(times^2) + I((times - median(times))^2 *
#>     (times > median(times))), data = mcycle)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -84.230 -25.431   2.499  29.490  85.089
#>
#> Coefficients:
#>                                                       Estimate
#> (Intercept)                                           88.98062
#> times                                                -16.91005
#> I(times^2)                                             0.47930
#> I((times - median(times))^2 * (times > median(times))) -0.61399
#>                                                      Std. Error t value
#> (Intercept)                                            20.22599   4.399
#> times                                                   2.37078  -7.133
#> I(times^2)                                              0.06379   7.514
#> I((times - median(times))^2 * (times > median(times)))  0.08967  -6.847
#>                                                      Pr(>|t|)
#> (Intercept)                                          2.25e-05 ***
#> times                                                6.30e-11 ***
#> I(times^2)                                           8.46e-12 ***
#> I((times - median(times))^2 * (times > median(times))) 2.76e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 38.74 on 129 degrees of freedom
#> Multiple R-squared:  0.372,  Adjusted R-squared:  0.3574
#> F-statistic: 25.47 on 3 and 129 DF,  p-value: 5.254e-13
summary(cub)
#>
#> Call:
#> lm(formula = accel ~ times + I(times^2) + I(times^3) + I((times -
#>     median(times))^3 * (times > median(times))), data = mcycle)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
```

```
#> -79.270 -29.451   2.896  26.939  98.702
#>
#> Coefficients:
#>                                                           Estimate
#> (Intercept)                                              42.228420
#> times                                                    -8.079708
#> I(times^2)                                                0.093281
#> I(times^3)                                                0.003401
#> I((times - median(times))^3 * (times > median(times))) -0.014646
#>                                                          Std. Error t value
#> (Intercept)                                               29.559740   1.429
#> times                                                      5.889478  -1.372
#> I(times^2)                                                 0.349112   0.267
#> I(times^3)                                                 0.006160   0.552
#> I((times - median(times))^3 * (times > median(times)))   0.008509  -1.721
#>                                                          Pr(>|t|)
#> (Intercept)                                                0.1556
#> times                                                      0.1725
#> I(times^2)                                                 0.7897
#> I(times^3)                                                 0.5818
#> I((times - median(times))^3 * (times > median(times)))   0.0876 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 39.7 on 128 degrees of freedom
#> Multiple R-squared:  0.3454, Adjusted R-squared:  0.325
#> F-statistic: 16.89 on 4 and 128 DF,  p-value: 3.847e-11
ggplot(mcycle, aes(x=times, y=accel)) +
  geom_point() +
  geom_line(mcycle, mapping=aes(x=times, y=predict(quad), color="Quad"))+
  geom_line(mcycle, mapping=aes(x=times, y=predict(cub), color="Cub"))
```

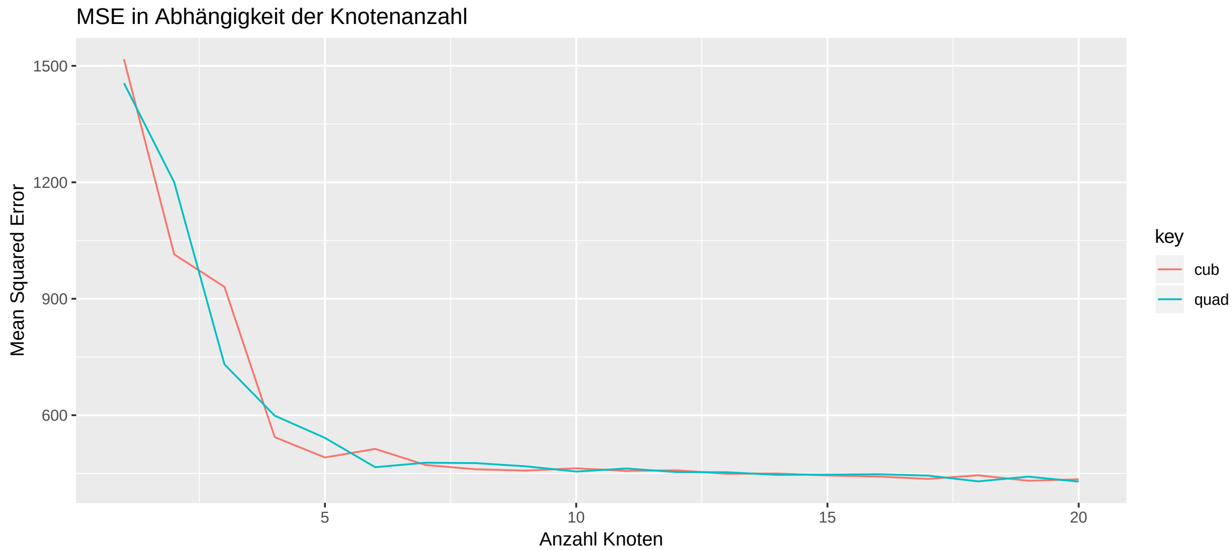


b)

```
compute_quantiles = function(i) {
  k = seq(0, 1, length.out=i+2)
  return (quantile(mcycle$times, probs = k[2:(length(k)-1)]))
}
k = 20
mse_quad = c()
mse_cub = c()
loocv=function(fit){
  h=lm.influence(fit)$h
  pred <- predict(fit)
  return(mean((pred-mcycle$accel)^2))
}
mse_quad = c()
mse_cub = c()
for(i in 1:k) {
  quad <- lm(mcycle$accel ~ bs(mcycle$times, degree = 2, knots = compute_quantiles(i)), data=mcycle)
  cub <- lm(mcycle$accel ~ bs(mcycle$times, degree = 3, knots = compute_quantiles(i)), data=mcycle)
  mse_quad <- append(mse_quad, loocv(quad))
  mse_cub <- append(mse_cub, loocv(cub))
}
mse <- NULL
mse$quad <- mse_quad
mse$cub <- mse_cub
mse <- as.data.frame(mse)
mse2 <- cbind(mse, seq(1:20))
names(mse2)[3] <- "index"
mse2 <- gather(mse2, key=key, value=value, -index)
mse2
#>    index  key     value
#> 1      1 quad 1455.4493
#> 2      2 quad 1200.0094
#> 3      3 quad  731.0319
#> 4      4 quad  598.8348
#> 5      5 quad  541.6380
#> 6      6 quad  466.1292
#> 7      7 quad  477.5380
#> 8      8 quad  476.6834
#> 9      9 quad  468.3124
#> 10    10 quad  454.9234
#> 11    11 quad  462.6994
#> 12    12 quad  453.5038
#> 13    13 quad  453.0113
#> 14    14 quad  446.4020
#> 15    15 quad  446.9456
#> 16    16 quad  447.9795
#> 17    17 quad  444.3495
#> 18    18 quad  429.5835
#> 19    19 quad  441.7136
#> 20    20 quad  428.9550
#> 21     1  cub 1516.9535
#> 22     2  cub 1014.3291
```

```r
#> 23     3   cub  930.6178
#> 24     4   cub  543.5729
#> 25     5   cub  491.1167
#> 26     6   cub  513.2543
#> 27     7   cub  471.8319
#> 28     8   cub  460.6924
#> 29     9   cub  457.3600
#> 30    10   cub  463.1924
#> 31    11   cub  456.4856
#> 32    12   cub  458.0485
#> 33    13   cub  449.0112
#> 34    14   cub  449.8635
#> 35    15   cub  444.8236
#> 36    16   cub  442.1034
#> 37    17   cub  435.9127
#> 38    18   cub  445.2312
#> 39    19   cub  431.1811
#> 40    20   cub  434.8586
gather(mse)
#>     key     value
#> 1  quad 1455.4493
#> 2  quad 1200.0094
#> 3  quad  731.0319
#> 4  quad  598.8348
#> 5  quad  541.6380
#> 6  quad  466.1292
#> 7  quad  477.5380
#> 8  quad  476.6834
#> 9  quad  468.3124
#> 10 quad  454.9234
#> 11 quad  462.6994
#> 12 quad  453.5038
#> 13 quad  453.0113
#> 14 quad  446.4020
#> 15 quad  446.9456
#> 16 quad  447.9795
#> 17 quad  444.3495
#> 18 quad  429.5835
#> 19 quad  441.7136
#> 20 quad  428.9550
#> 21  cub 1516.9535
#> 22  cub 1014.3291
#> 23  cub  930.6178
#> 24  cub  543.5729
#> 25  cub  491.1167
#> 26  cub  513.2543
#> 27  cub  471.8319
#> 28  cub  460.6924
#> 29  cub  457.3600
#> 30  cub  463.1924
#> 31  cub  456.4856
#> 32  cub  458.0485
#> 33  cub  449.0112
```

```r
#> 34  cub  449.8635
#> 35  cub  444.8236
#> 36  cub  442.1034
#> 37  cub  435.9127
#> 38  cub  445.2312
#> 39  cub  431.1811
#> 40  cub  434.8586
ggplot(aes(x=index, y=value, col=key), data = mse2) +
  geom_line() +
  xlab("Anzahl Knoten") +
  ylab("Mean Squared Error") +
  ggtitle("MSE in Abhängigkeit der Knotenanzahl")
```



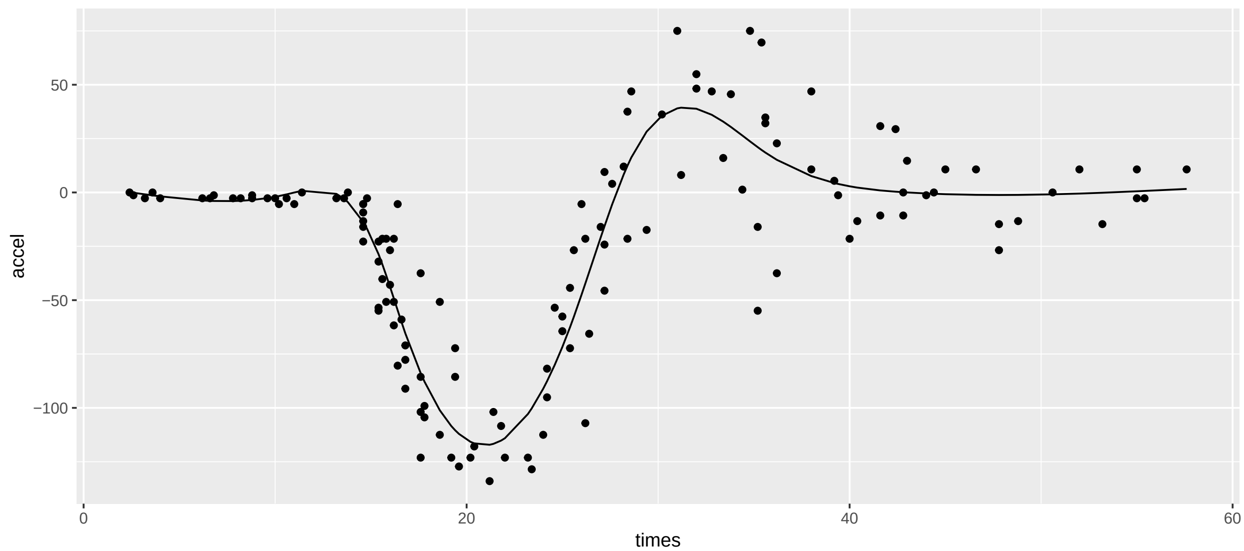
Ab 5 Knoten scheint sich der Mean Squared Error nicht mehr signifikant zu verbessern, die optimale Anzahl an Knoten würden wir deshalb auf 5 festlegen.

c)

```r
compute_params <- function(d, K){
  return(d+K+1)
}
quad_1_params <- compute_params(2, 1)
quad_5_params <- compute_params(2, 5)
cub_1_params <- compute_params(3, 1)
cub_5_params <- compute_params(3, 5)
quad_1_params
#> [1] 4
quad_5_params
#> [1] 8
cub_1_params
#> [1] 5
cub_5_params
#> [1] 9
```

d)

```
ns_reg <- lm(mcycle$accel ~ ns(mcycle$times, df = 9), data=mcycle)
ggplot(aes(x=times, y=accel), data=mcycle) +
  geom_point() +
  geom_line(aes(x=times, y=predict(ns_reg)), data=mcycle)
```



Aufgabe 1 (Alternativ)

```
df = mcycle
x = df$times
y = df$accel
```

a)

```
med = median(df$times)
knots = c(med)
```

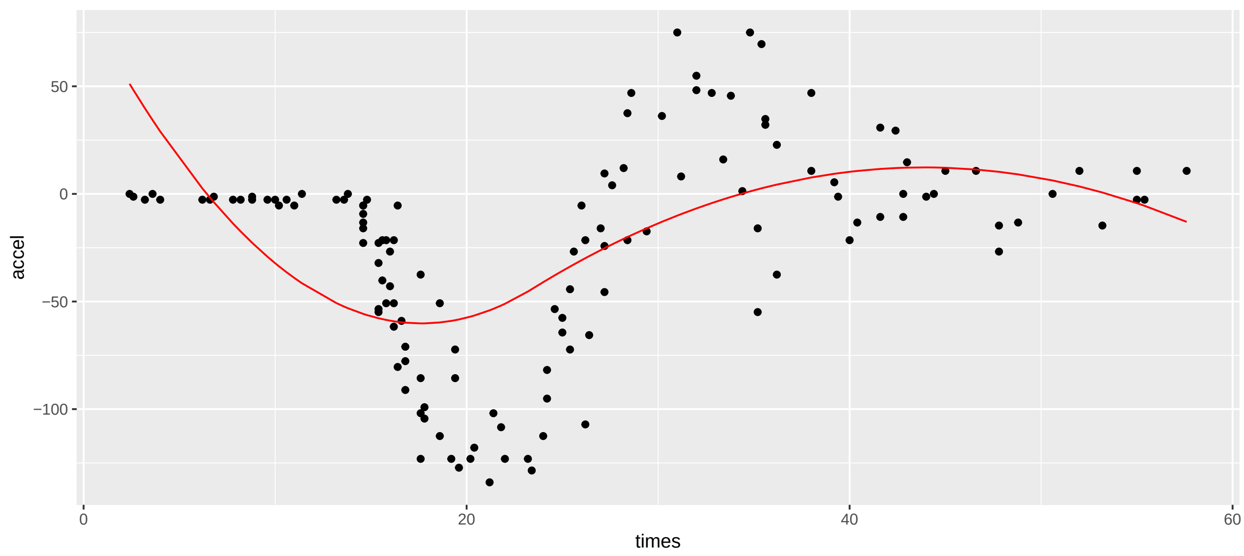
Quadratisch

```
h = function(x, knots, d, i) {
  if (i <= d + 1) {
    out = x**(i-1)
    return(out)
  }
  else {
    out = (x - knots[i-d-1])
    out = ifelse(out < 0, 0, out)
    out = out ** d
```

```
    return(out)
  }
}
Z = function(X, knots, d) {
  h1 = h(X, knots, d, 1)
  h2 = h(X, knots, d, 2)
  h3 = h(X, knots, d, 3)
  h4 = h(X, knots, d, 4)
  v = c(h1, h2, h3, h4)
  out = matrix(
    data = v,
    nrow = length(h1),
    ncol = 4
  )
  return(out)
}
transpose = function(M) sapply(1:nrow(M), function(i) M[i,])
X = Z(x, knots, 2)
XT = transpose(X)
hatb = ginv(XT %*% X) %*% XT %*% y
f = function(x, hatb, knots, d) {
  h1 = h(x, knots, d, 1)
  h2 = h(x, knots, d, 2)
  h3 = h(x, knots, d, 3)
  h4 = h(x, knots, d, 4)
  v = c(h1, h2, h3, h4)
  out = hatb[1] * h1 + hatb[2] * h2 + hatb[3] * h3 + hatb[4] * h4
  return(out)
}
f2 = function(x) {
  return(f(x, hatb, knots, 2))
}
haty = f(x, hatb, knots, 2)
df2 = data.frame(x = x, y = haty)
```

Plot:

```
gg = ggplot(
  data = df,
  mapping = aes(
    x = times,
    y = accel
  )
)
gg = gg + geom_point()
gg + geom_line(
  data = df2,
  color = 'red',
  aes(x = x, y = y)
)
```



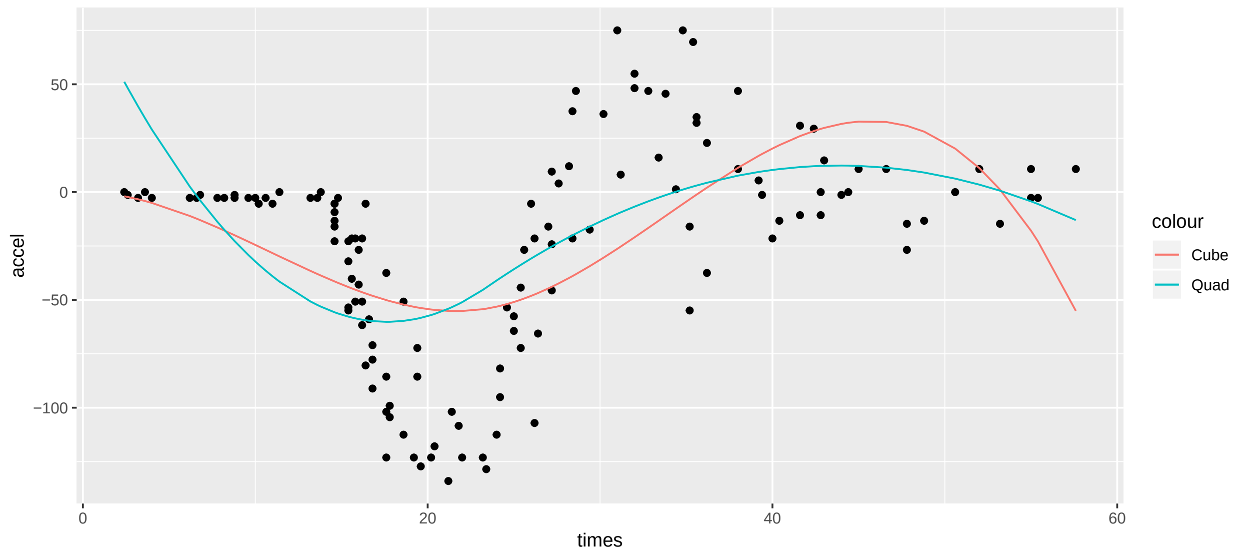
Kubisch

```
h = function(x, knots, d, i) {
  if (i <= d + 1) {
    out = x**(i-1)
    return(out)
  }
  else {
    out = (x - knots[i-d-1])
    out = ifelse(out < 0, 0, out)
    out = out ** d
    return(out)
  }
}
Z = function(X, knots, d) {
  h1 = h(X, knots, d, 1)
  h2 = h(X, knots, d, 2)
  h3 = h(X, knots, d, 3)
  h4 = h(X, knots, d, 4)
  h5 = h(X, knots, d, 5)
  v = c(h1, h2, h3, h4, h5)
  out = matrix(
    data = v,
    nrow = length(h1),
    ncol = 5
  )
  return(out)
}
transpose = function(M) sapply(1:nrow(M), function(i) M[i,])
X = Z(x, knots, 3)
XT = transpose(X)
hatb = ginv(XT %*% X) %*% XT %*% y
f = function(x, hatb, knots, d) {
  h1 = h(x, knots, d, 1)
  h2 = h(x, knots, d, 2)
```

```
  h3 = h(x, knots, d, 3)
  h4 = h(x, knots, d, 4)
  h5 = h(x, knots, d, 5)
  v = c(h1, h2, h3, h4, h5)
  out = hatb[1] * h1 + hatb[2] * h2 + hatb[3] * h3 + hatb[4] * h4 + hatb[5] * h5
  return(out)
}
f3 = function(x) {
  return(f(x, hatb, knots, 3))
}
haty = f(x, hatb, knots, 3)
df3 = data.frame(x = x, y = haty)
```

Plot:

```
gg = ggplot(
  data = df,
  mapping = aes(
    x = times,
    y = accel
  )
)
gg = gg + geom_point()
gg + geom_line(
  data = df3,
  aes(x = x, y = y, color = "Cube")
) + geom_line(
  data = df2,
  aes(x = x, y = y, color = "Quad")
)
```



c)

Die Anzahl an freien Parametern wird berechnet mittels $d + K + 1$ mit d Grad und K Knotenanzahl.

A2

a)

Sei $[a, b]$ das Regressionsintervall, dann wird ein linearer Anfang und Abgang gefordert. Sprich es folgen die Bedingungen:

$$f''(a) = 0 \tag{1}$$
$$f''(b) = 0 \tag{2}$$

b)

Seien $f''(a) = f''(b) = 0$. Zunächst leiten wird ab.

$$f'(x) = \sum_{j=1}^{3} j a_j x^{j-1} + 3 \sum_{k=1}^{K} b_k (x - \xi_k)_+^2 \tag{3}$$
$$f''(x) = 2a_2 + 6a_3 x + 6 \sum_{k=1}^{K} b_k (x - \xi_k)_+ \tag{4}$$

Nun setzen wir die Bedingungen ein. Wir nutzen aus, dass grundsätzlich $(a - \xi_k)_+ = 0$ und $(b - \xi_k)_+ > 0$ gilt. Dies folgt aus $a < \xi_1 < \cdots < \xi_K < b$.

$$f''(a) = 2a_2 + 6a_3 a + 6 \sum_{k=1}^{K} b_k (a - \xi_k)_+ = 0 \tag{5}$$
$$\implies a_2 + 3a_3 a = 0 \implies a_2 = a_3 = 0 \tag{6}$$

Alternativ könnte man auch $a_2 = -3a_3$ fordern, aber da wir die Wahl haben, nehmen wir die einfache Bedingung. Ferner folgt nun:

$$f''(b) = \sum_{k=1}^{K} b_k (b - \xi_k)_+ = 0 \tag{7}$$
$$\implies \sum_{k=1}^{K} b_k = \frac{1}{b} \sum_{k=1}^{K} b_k \xi_k \implies \sum_{k=1}^{K} b_k = \sum_{k=1}^{K} b_k \xi_k = 0 \tag{8}$$

Für alle k ist $\frac{\xi_k}{b} \in (0, 1)$ wodurch obige Gleichung nur dann gelten kann, wenn beide Summen Null sind.

c)

Ein kubischer Spline mit K Knoten hat $K+4$ freie Parameter. Nun haben wir vier weitere Nebenbedingungen, wodurch wir vier Freiheitsgrade verlieren und folglich haben wir nur noch K freie Parameter.

d)

Ohne viel aufzuschreiben macht man sich klar, dass dass die Ableitungen einfach sind insofern, dass die Potenzen aggregiert als 6 vor die Summe verschwinden. Weiter nutzt man, dass $(a-\xi_k)_+ = 0$ und $(b-\xi_k)_+ > 0$ gilt und wir erhalten:

$$f''(a) = 6\sum_{k=1}^{K-2} \tilde{b}_k \cdot 0 = 0 \tag{9}$$

$$f''(b) = 6\sum_{k=1}^{K-2} \left(\frac{b - \xi_k - b + \xi_K}{\xi_K - \xi_k} - \frac{b - \xi_{K-1} - b + \xi_K}{\xi_K - \xi_{K-1}} \right) == 6\sum_{k=1}^{K-2} (1-1) = 0 \tag{10}$$

Folglich lässt sich jeder natürliche kubische Spline über die angegebene Basiserweiterung darstellen.