# NLNP Praktikum 8

*Robin Baudisch, Merlin Kopfmann, Maximilian Neudert*

## Inhaltsverzeichnis

**A1** **2**
a) . . . . . 2
b) . . . . . 4
c) . . . . . 6
d) . . . . . 6

**A1 (Alternativ)** **7**
a) . . . . . 7
c) . . . . . 10

**A2** **10**
a) . . . . . 10
b) . . . . . 10
c) . . . . . 11
d) . . . . . 11

# A1

## a)

```
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 = mcycle)
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)
```

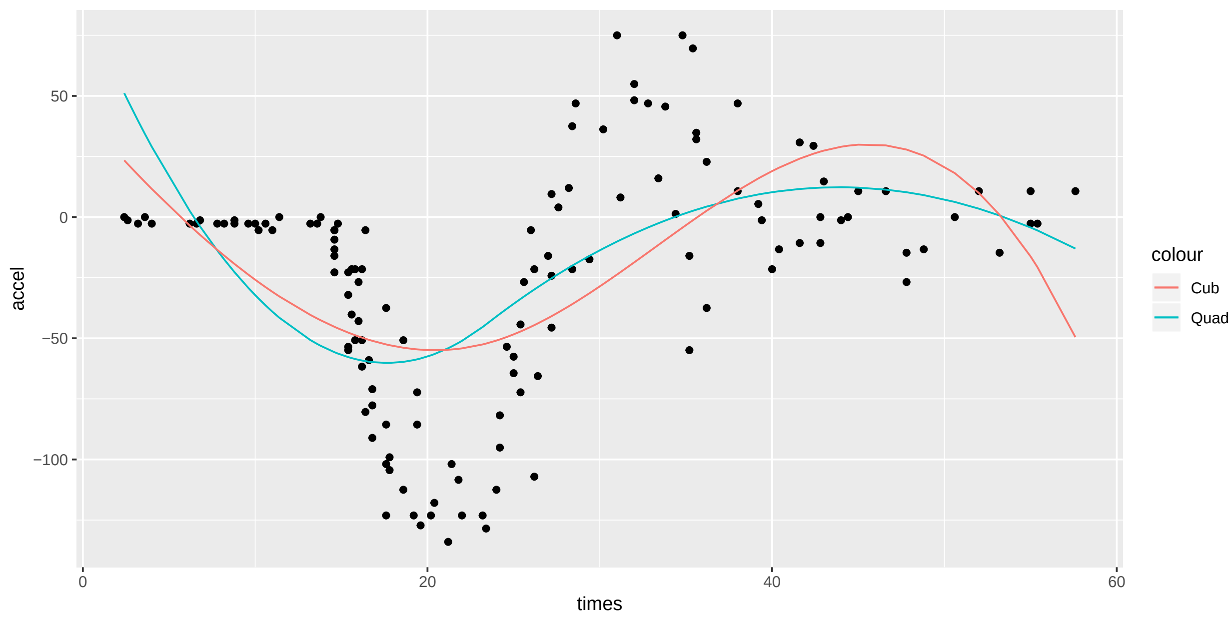
```
Call:
lm(formula = accel ~ times + I(times^2) + I(times^3) + I((times -
    median(times))^3 * (times > median(times))), data = mcycle)
```

```
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
```

```r
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"))
```



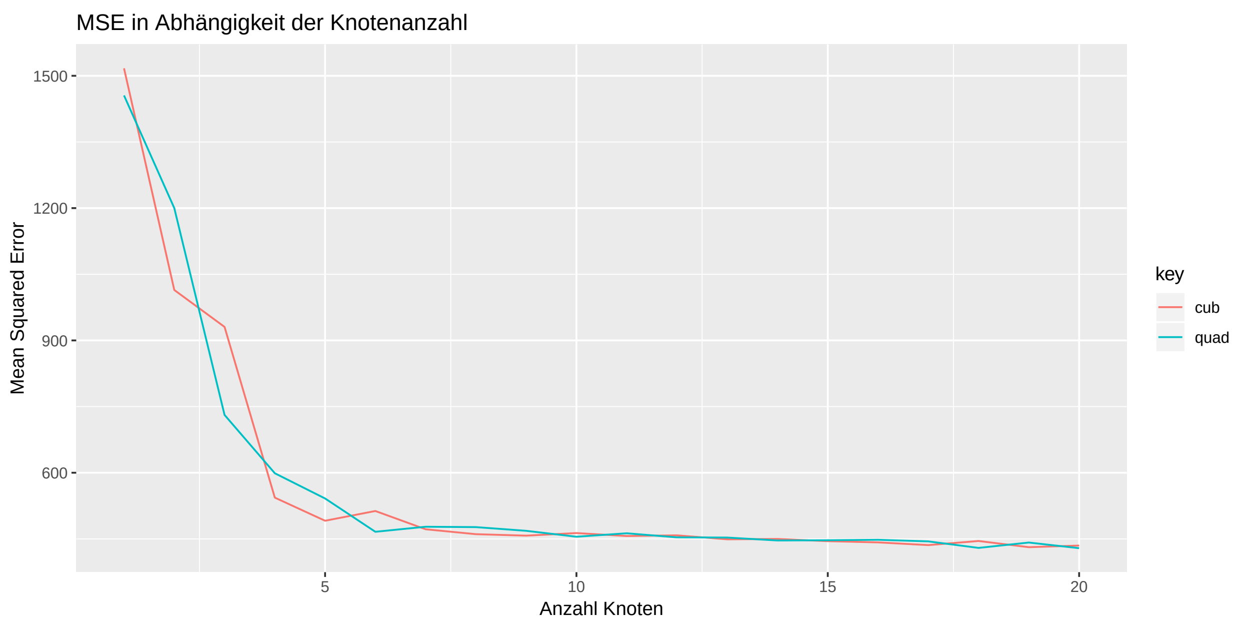

**b)**

```
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
```

| quad | cub | index |
|---:|---:|---:|
| 1455.4493 | 1516.9535 | 1 |
| 1200.0094 | 1014.3291 | 2 |
| 731.0319 | 930.6178 | 3 |
| 598.8348 | 543.5729 | 4 |
| 541.6380 | 491.1167 | 5 |
| 466.1292 | 513.2543 | 6 |
| 477.5380 | 471.8319 | 7 |
| 476.6834 | 460.6924 | 8 |
| 468.3124 | 457.3600 | 9 |
| 454.9234 | 463.1924 | 10 |
| 462.6994 | 456.4856 | 11 |
| 453.5038 | 458.0485 | 12 |
| 453.0113 | 449.0112 | 13 |
| 446.4020 | 449.8635 | 14 |
| 446.9456 | 444.8236 | 15 |
| 447.9795 | 442.1034 | 16 |
| 444.3495 | 435.9127 | 17 |
| 429.5835 | 445.2312 | 18 |
| 441.7136 | 431.1811 | 19 |
| 428.9550 | 434.8586 | 20 |




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.

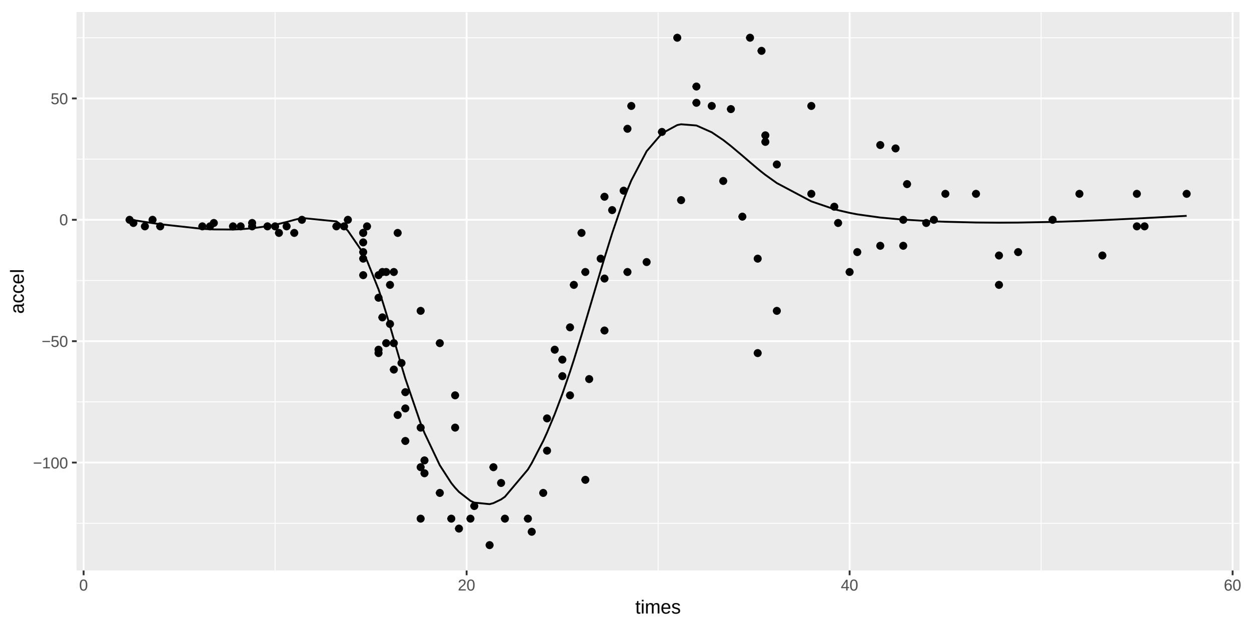
**c)**

```
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)
val = c(quad_1_params, quad_5_params, cub_1_params, cub_5_params)
key = c("quad_1", "quad_5", "cub_1", "cub_5")
df = data.frame(model = key, nparams = val)
df
```

| model | nparams |
|---|---|
| quad_1 | 4 |
| quad_5 | 8 |
| cub_1 | 5 |
| cub_5 | 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)
```



## A1 (Alternativ)

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

### a)

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

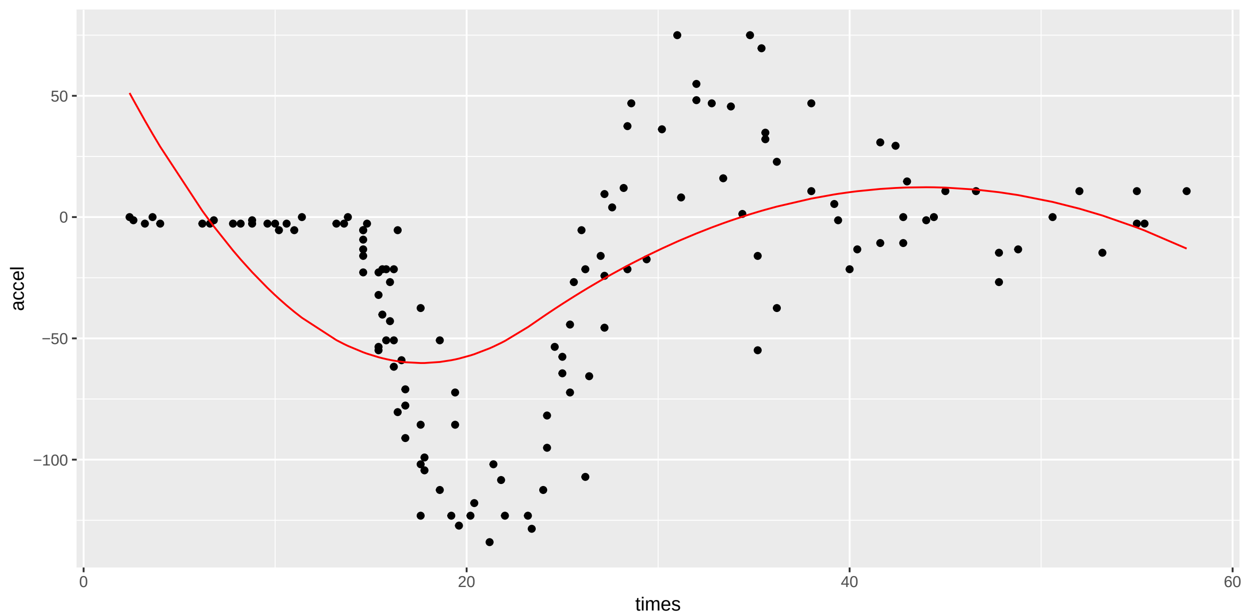
#### Quadratisch

```r
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))
```



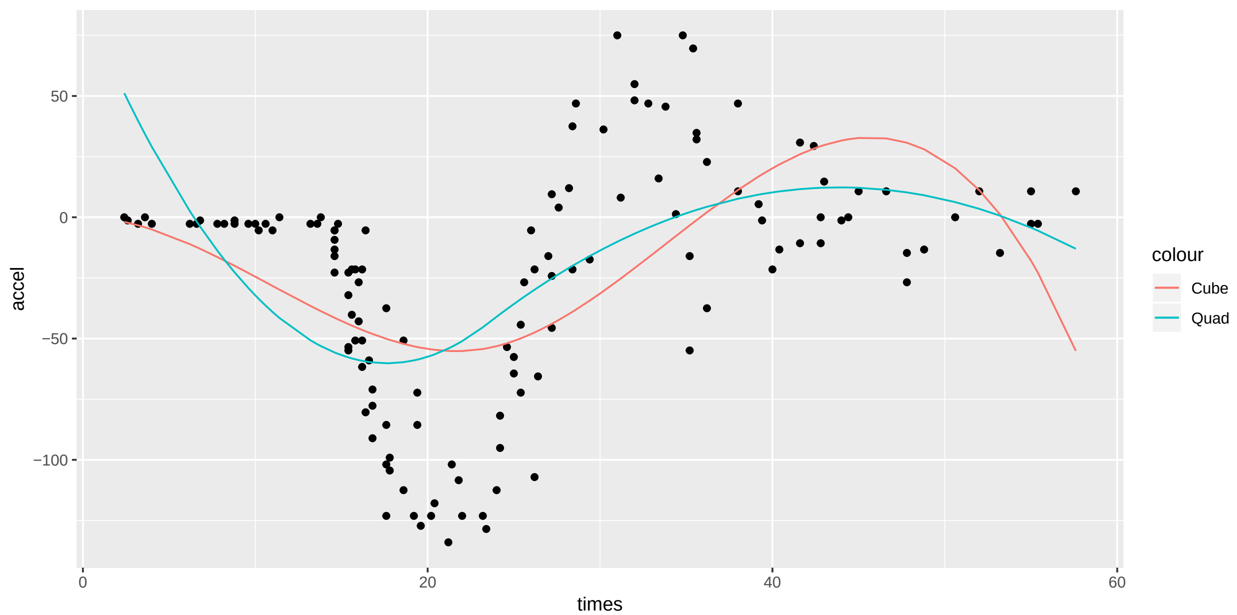

**Kubisch**

```
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:

$$f''(x) = 0, \quad x \in [a, \xi_i] \tag{1}$$
$$f''(x) = 0, \quad x \in [\xi_K, b] \tag{2}$$

### 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 $a < \xi_1 < \cdots < \xi_K < b$.

$$f''(x) = 2a_2 + 6a_3x + 6\sum_{k=1}^{K} b_k(x-\xi_k)_+ = 0 \tag{5}$$

$$\implies a_2 + 3a_3x = 0, \quad \forall x \in [a, \xi_i] \tag{6}$$

$$\implies a_2 = a_3 = 0 \tag{7}$$

Ferner erhalten wir nun:

$$f''(x) = \sum_{k=1}^{K} b_k(x-\xi_k)_+ = 0, \quad x \in [\xi_K, b] \tag{8}$$

$$\implies \sum_{k=1}^{K} b_k = \frac{1}{x}\sum_{k=1}^{K} b_k\xi_k, \quad x \in [\xi_K, b] \tag{9}$$

$$\implies \sum_{k=1}^{K} b_k = \sum_{k=1}^{K} b_k\xi_k = 0 \tag{10}$$

**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)**

Um dies zu zeigen müssen wir prüfen, dass $f(x)$ die Eigenschaften aus „b)“ erfüllt. $a_2 = a_3 = 0$ sieht man schnell. Die anderen beiden Eigenschaften können wir durch ausmultiplizieren nachprüfen. Dies bleibt dem aufmerksamen Leser als Übungsaufgabe überlassen.