Examples of fitting various piecewise-continuous functions to data, using basis functions in doing the regressions.

Change log:

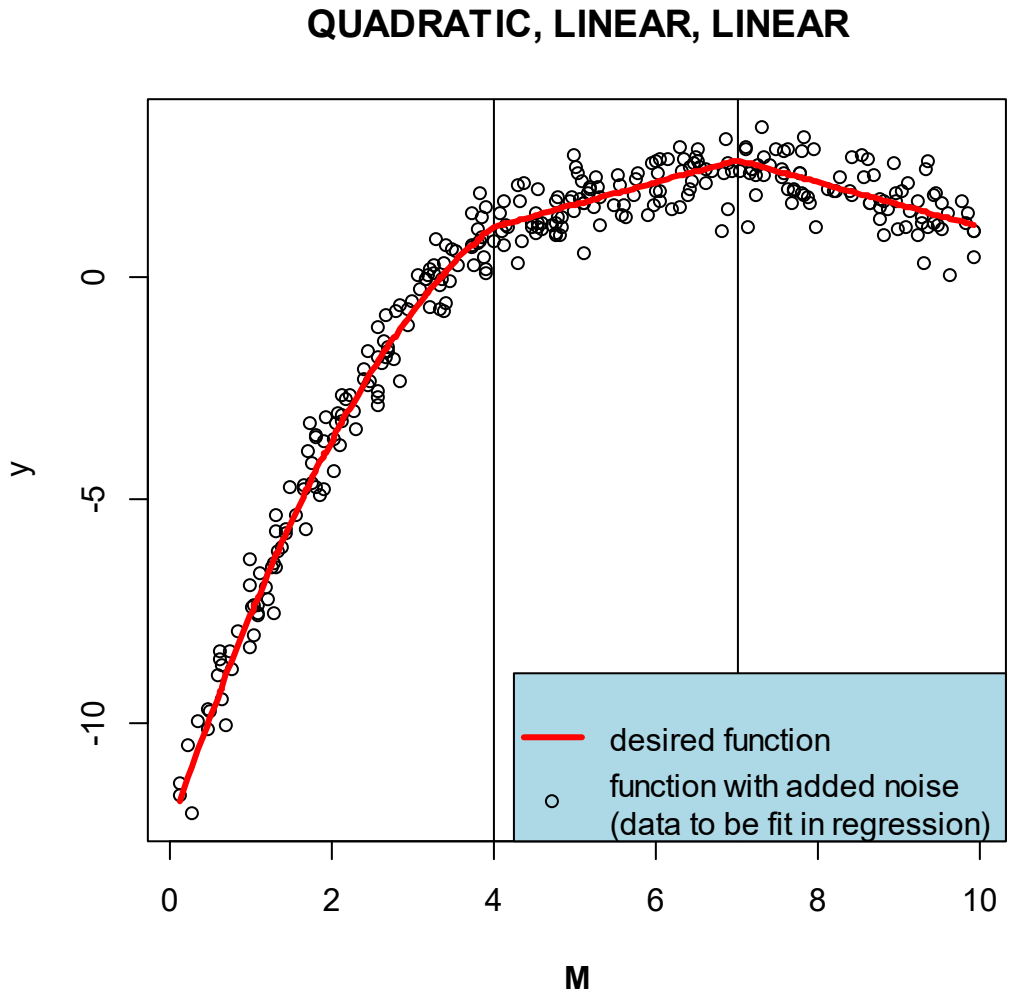
01nov23: minor revisions of 16jul23 version: I had left out the basis functions for the first example; a few small changes in text

David M. Boore

These examples in this document used R to do the regression. See also *Notes_on_piecewise_continuous_regression.doc* for more detail on why the basis functions used below guarantee continuity at the breakpoints.

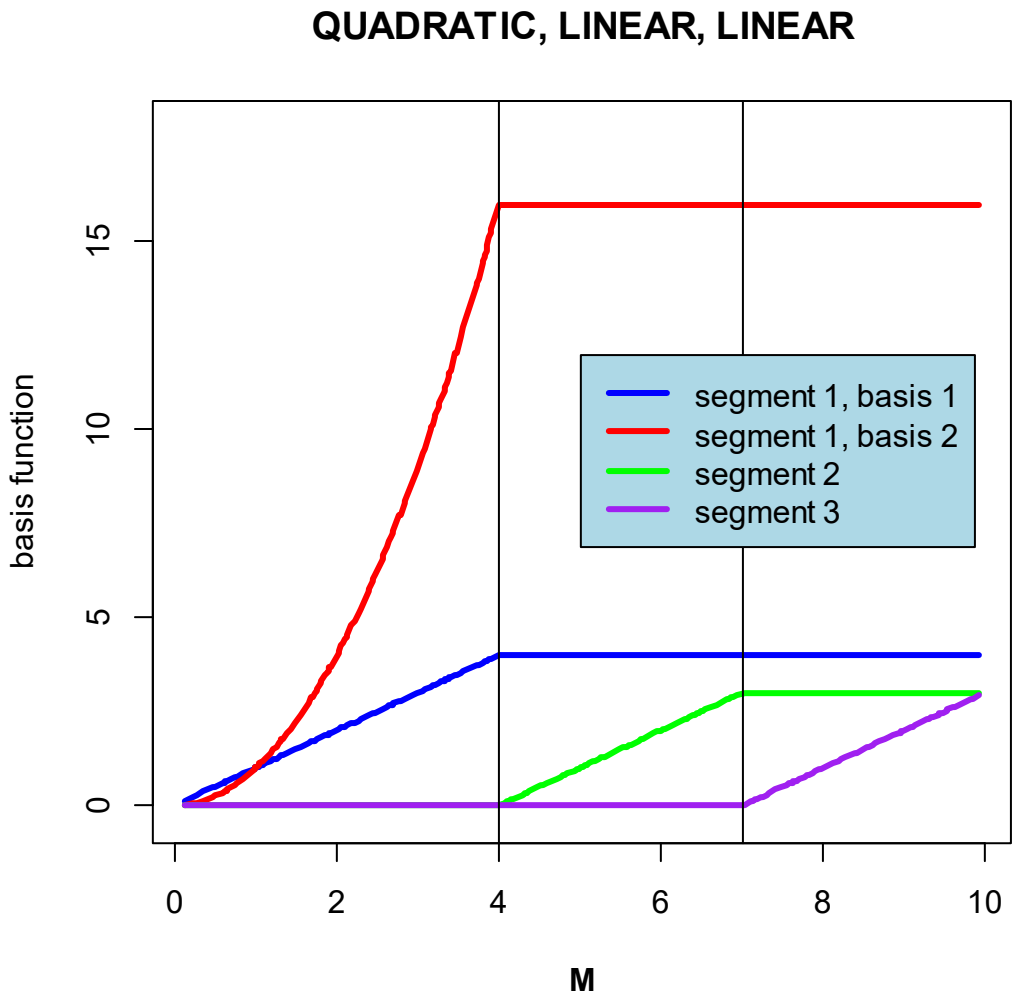
```
###################################
# Based on Eric Thompson's program piecewise-continuous.R, using DMB basis
functions

#-------------------------------------------------------------------------
-----------

#-------------------------------------------------------------------------
-----------
Title <- "QUADRATIC, LINEAR, LINEAR"
# Define function
# quadratic, M<4, linear, M 4 to 7, linear M>7

# Breakpoints:
c <- c(4, 7)
y <- function(M,c){
  ifelse(M<c[1],1.13+1.4*(M-c[1])-0.5*(M-c[1])^2,
    ifelse(M<=c[2],1.13+0.5*(M-c[1]), 1.13+0.5*(c[2]-c[1])-0.5*(M-c[2])))
 }

#Generate data
set.seed(1)
n <- 300
M <- runif(n, 0, 10)
Msorted <- sort(M)

# Add some noise:
yn <- y(M,c) + rnorm(n, sd = 0.5)

plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)
```

```
abline(v = c)
lines(Msorted, y(Msorted,c),lwd=3, col="red")
legend("bottomright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)"),
        lty=c(1,NA), lwd=c(3,NA), pch=c(NA,1), col=c("red", "black"),
bg="lightblue")
```



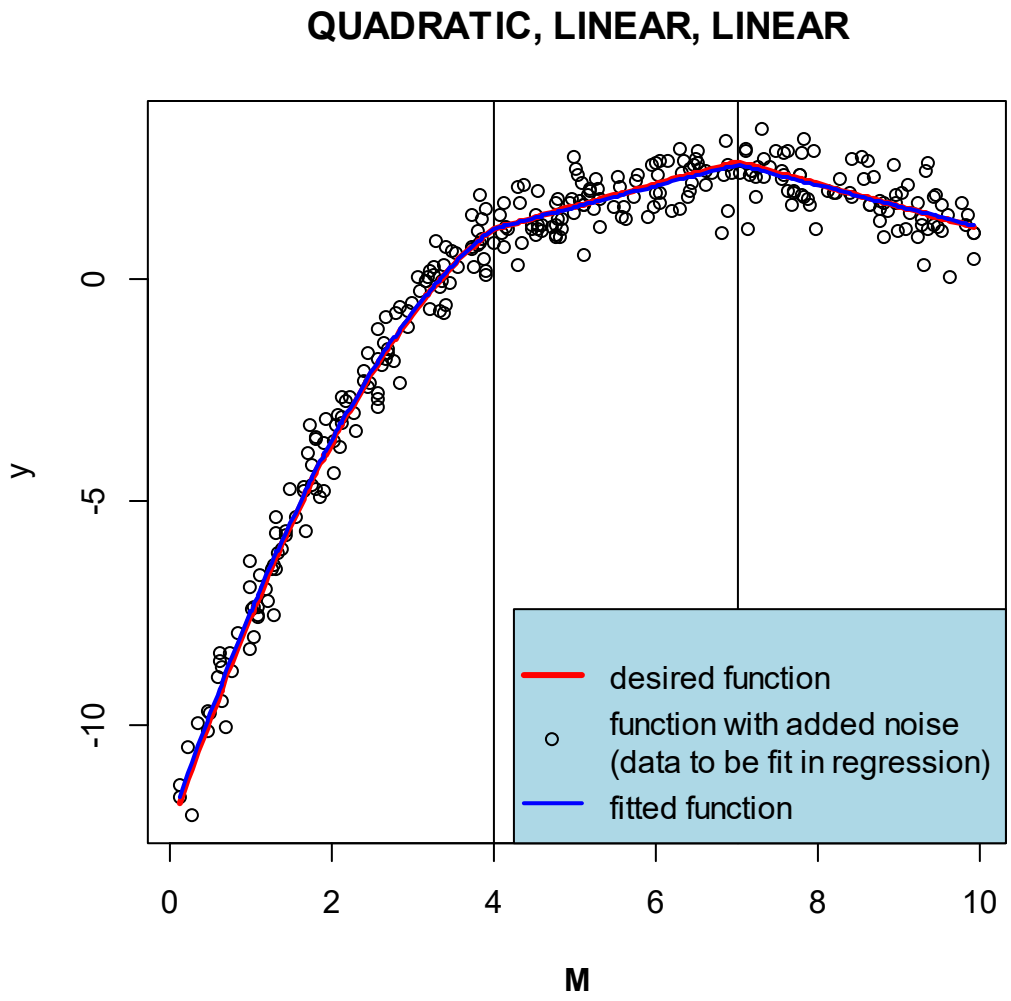

```
plot(Msorted, b1(Msorted,c[1]), xlab=substitute(paste(bold("M"))),
ylab="basis function",
  ylim=c(-0.25,18.0), type="l", col = "blue", lwd=3, main=Title)
lines(Msorted, b1.2(Msorted,c[1]), col = "red", lwd=3)
lines(Msorted, b2(Msorted,c[1],c[2]), col = "green", lwd=3)
lines(Msorted, b3(Msorted,c[2]), col = "purple", lwd=3)
abline(v = c)
```

```
legend(5,12, legend=c("segment 1, basis 1","segment 1, basis 2", "segment 2",
"segment 3"),
        lty=rep(1,4), lwd=rep(3,4), col=c("blue", "red", "green", "purple"),
bg="lightblue")


# Compute basis function for each region:
b1 <- function(x,R){ifelse(x<R,x,R)}
b1.2 <- function(x,R){ifelse(x<R,x^2,R^2)}
b2 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,x-L,R-L))}
b3 <- function(x,L){ifelse(x<L,0,x-L)}
# NOTE: L and R are not the necessarily the same for all segments; they are
the breakpoints on either side of each segment.
# See the call to lm to see what values of L and R are used for each basis
function.

plot(Msorted, b1(Msorted,c[1]), xlab=substitute(paste(bold("M"))),
ylab="basis function",
  ylim=c(-0.25,18.0), type="l", col = "blue", lwd=3, main=Title)
lines(Msorted, b1.2(Msorted,c[1]), col = "red", lwd=3)
lines(Msorted, b2(Msorted,c[1],c[2]), col = "green", lwd=3)
lines(Msorted, b3(Msorted,c[2]), col = "purple", lwd=3)
abline(v = c)
legend(5,12, legend=c("segment 1, basis 1","segment 1, basis 2", "segment 2",
"segment 3"),
        lty=rep(1,4), lwd=rep(3,4), col=c("blue", "red", "green", "purple"),
bg="lightblue")
```

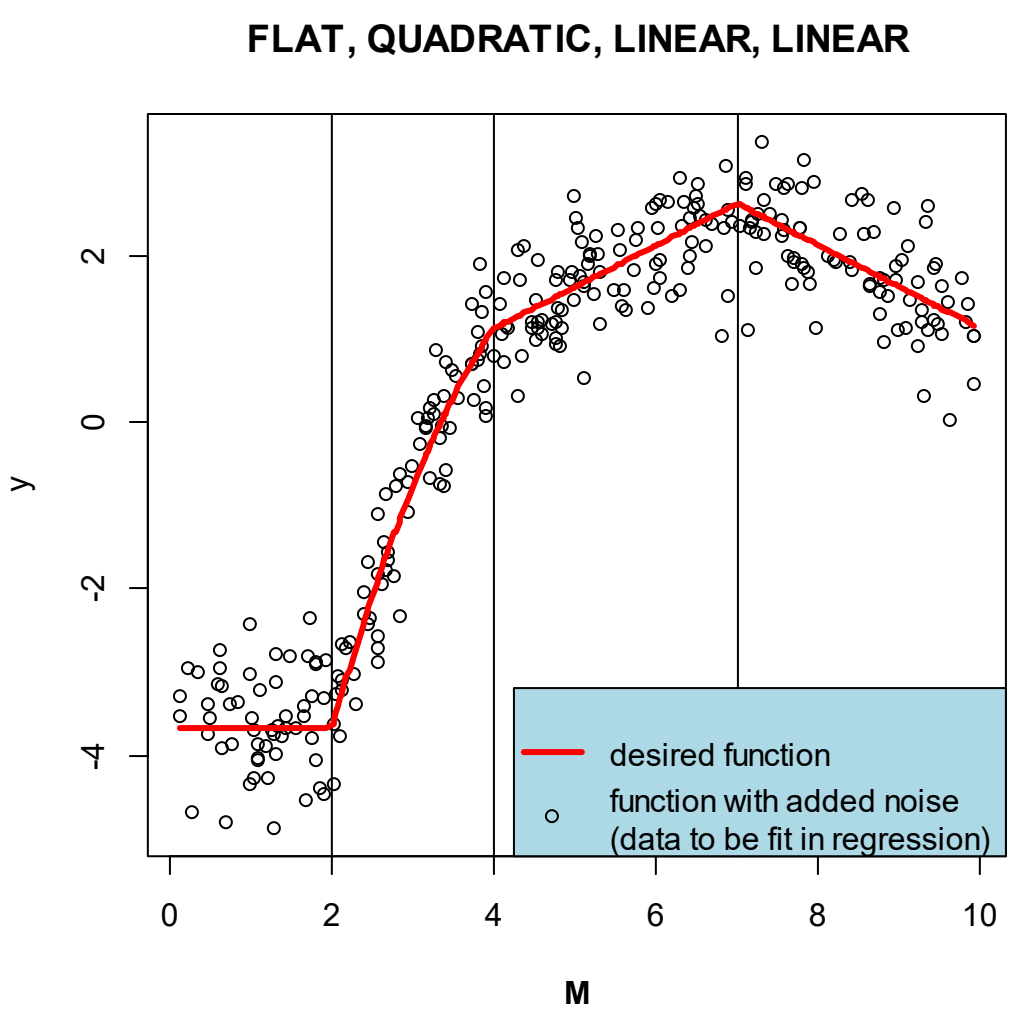



```
(NOTE: new figure made on 2022-10-18 using ylim chosen to show the b1.2
basis)

Model <- lm(yn ~ b1(M,c[1]) + b1.2(M,c[1]) + b2(M,c[1],c[2]) +  b3(M,c[2]) )
# Regression summary:
summary(Model)
#Call:
#lm(formula = yn ~ b1(M, c[1]) + b1.2(M, c[1]) + b2(M, c[1], c[2]) +
#    b3(M, c[2]))
#
#Residuals:
#     Min       1Q   Median       3Q      Max
#-1.44527 -0.30024 -0.01794  0.34079  1.25788
#
#Coefficients:
```

```
#                   Estimate Std. Error t value Pr(>|t|)
#(Intercept)       -12.31402    0.17147 -71.814   <2e-16 ***
#b1(M, c[1])        5.33851    0.16617  32.127   <2e-16 ***
#b1.2(M, c[1])     -0.49422    0.03438 -14.374   <2e-16 ***
#b2(M, c[1], c[2])  0.47826    0.04176  11.454   <2e-16 ***
#b3(M, c[2])       -0.46825    0.05141  -9.109   <2e-16 ***
#---
#Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
#Residual standard error: 0.5128 on 295 degrees of freedom
#Multiple R-squared:  0.9795,    Adjusted R-squared:  0.9793
#F-statistic:  3529 on 4 and 295 DF,  p-value: < 2.2e-16#Coefficients:

coef <- coefficients(Model)
#> coef
#      (Intercept)       b1(M, c[1])     b1.2(M, c[1]) b2(M, c[1], c[2])
b3(M, c[2])
#      -12.3140225         5.3385115        -0.4942249         0.4782630
-0.4682550

# Replot data
plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)
# Plot actual function
abline(v = c)
lines(Msorted,y(Msorted,c), lwd=3, col="red")

# Add predictions
yp <- coef[1]+coef[2]*b1(Msorted,c[1])+coef[3]*b1.2(Msorted,c[1]) +
      coef[4]*b2(Msorted,c[1],c[2]) +
      coef[5]*b3(Msorted,c[2])
lines(Msorted,yp, lwd=2, col="blue")
legend("bottomright", legend=c("desired function", "function with added
  noise\n(data to be fit in regression)", "fitted function"),
  lty=c(1,NA,1), lwd=c(3,NA,2), pch=c(NA,1,NA), col=c("red", "black",
  "blue"), bg="lightblue")
```

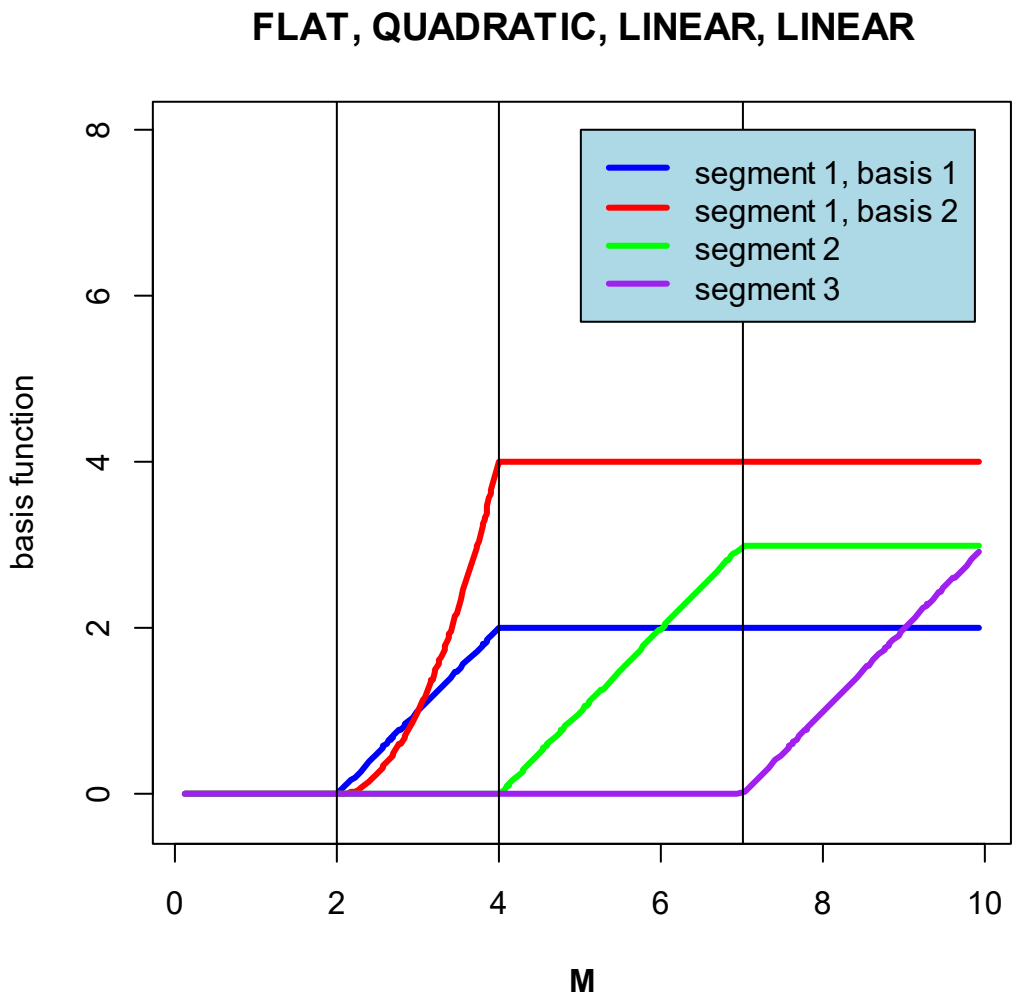

QUADRATIC, LINEAR, LINEAR
y
0
-5
-10
0
2
4
6
8
10
M
desired function
function with added noise
(data to be fit in regression)
fitted function

```
Title <- "FLAT, QUADRATIC, LINEAR, LINEAR"
# A basis function is not needed for a flat segment
# Define function

# Breakpoints:
c <- c(2, 4, 7)
y <- function(M,c){
  ifelse(M<c[1], -3.67,
    ifelse(M<c[2],1.13+1.4*(M-c[2])-0.5*(M-c[2])^2,
      ifelse(M<=c[3],1.13+0.5*(M-c[2]), 1.13+0.5*(c[3]-c[2])-0.5*(M-c[3]))))
 }


#Generate data
set.seed(1)
n <- 300
M <- runif(n, 0, 10)
Msorted <- sort(M)

# Add some noise:
yn <- y(M,c) + rnorm(n, sd = 0.5)

plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)
abline(v = c)
lines(Msorted, y(Msorted,c),lwd=3, col="red")
legend("bottomright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)"),
        lty=c(1,NA), lwd=c(3,NA), pch=c(NA,1), col=c("red", "black"),
bg="lightblue")
```



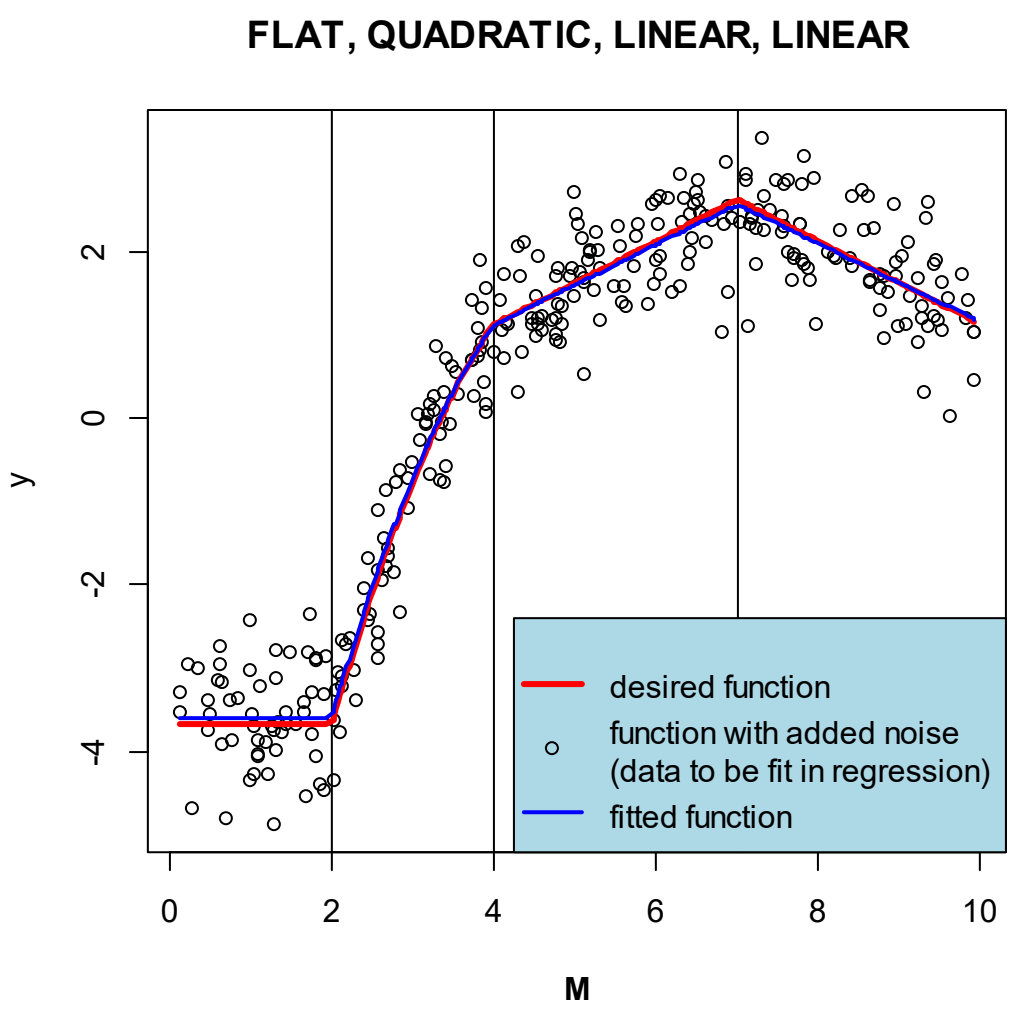

```
# Compute basis functions for each segment (a basis function is not needed
for the first segment, as the intercept
# in the regression will use the data in that segment):
b1.1 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,x-L,R-L))}
b1.2 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,(x-L)^2,(R-L)^2))}
b2 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,x-L,R-L))}
b3 <- function(x,L){ifelse(x<L,0,x-L)}

plot(Msorted, b1.1(Msorted,c[1],c[2]), xlab=substitute(paste(bold("M"))),
ylab="basis function",
  ylim=c(-0.25,8.0), type="l", col = "blue", lwd=3, main=Title)
lines(Msorted, b1.2(Msorted,c[1],c[2]), col = "red", lwd=3)
lines(Msorted, b2(Msorted,c[2],c[3]), col = "green", lwd=3)
lines(Msorted, b3(Msorted,c[3]), col = "purple", lwd=3)
abline(v = c)
```

```
legend(5,8, legend=c("segment 1, basis 1","segment 1, basis 2", "segment 2",
"segment 3"),
        lty=rep(1,4), lwd=rep(3,4), col=c("blue", "red", "green", "purple"),
bg="lightblue")
```

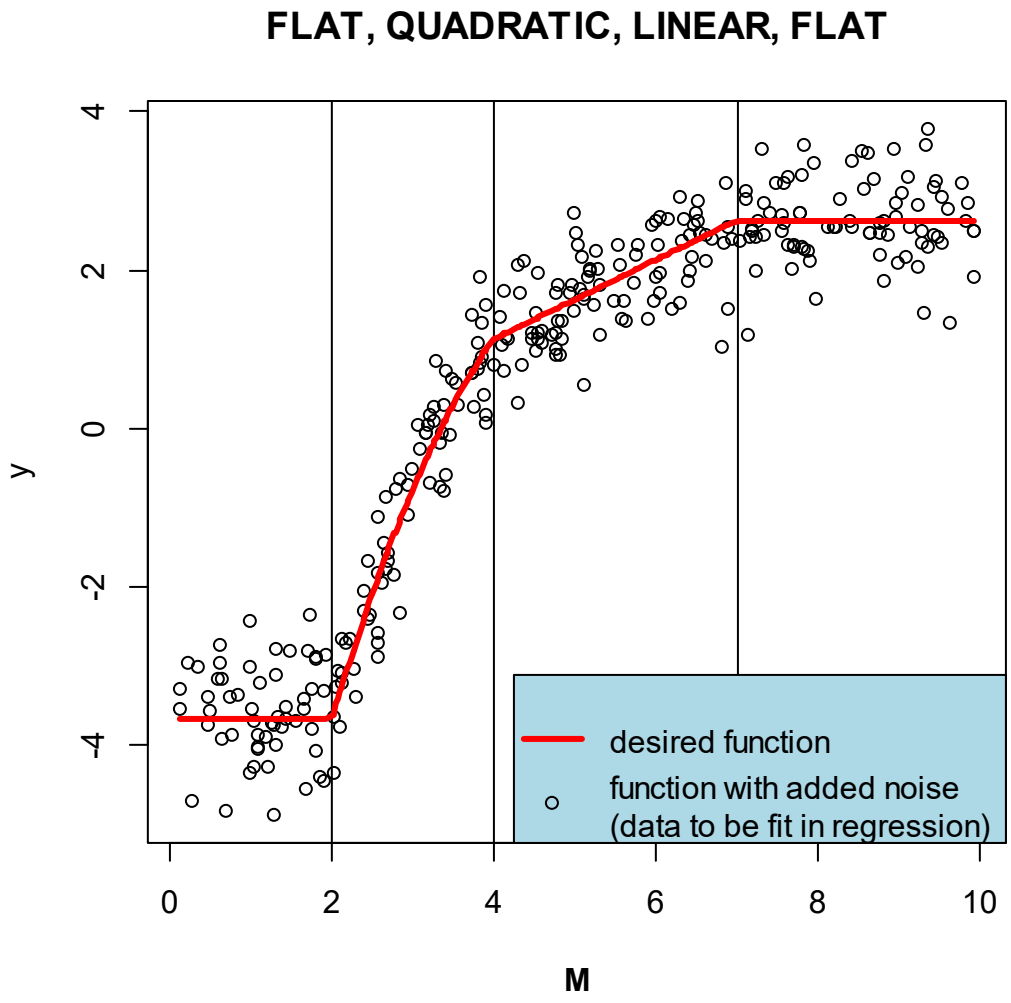



```
Model <- lm(yn ~ b1.1(M,c[1],c[2]) + b1.2(M,c[1],c[2]) + b2(M,c[2],c[3]) +
b3(M,c[3]))

# Regression summary:
summary(Model)

#Call:
#lm(formula = yn ~ b1.1(M, c[1], c[2]) + b1.2(M, c[1], c[2]) +
#    b2(M, c[2], c[3]) + b3(M, c[3]))
#
#Residuals:
#     Min       1Q   Median       3Q      Max
```

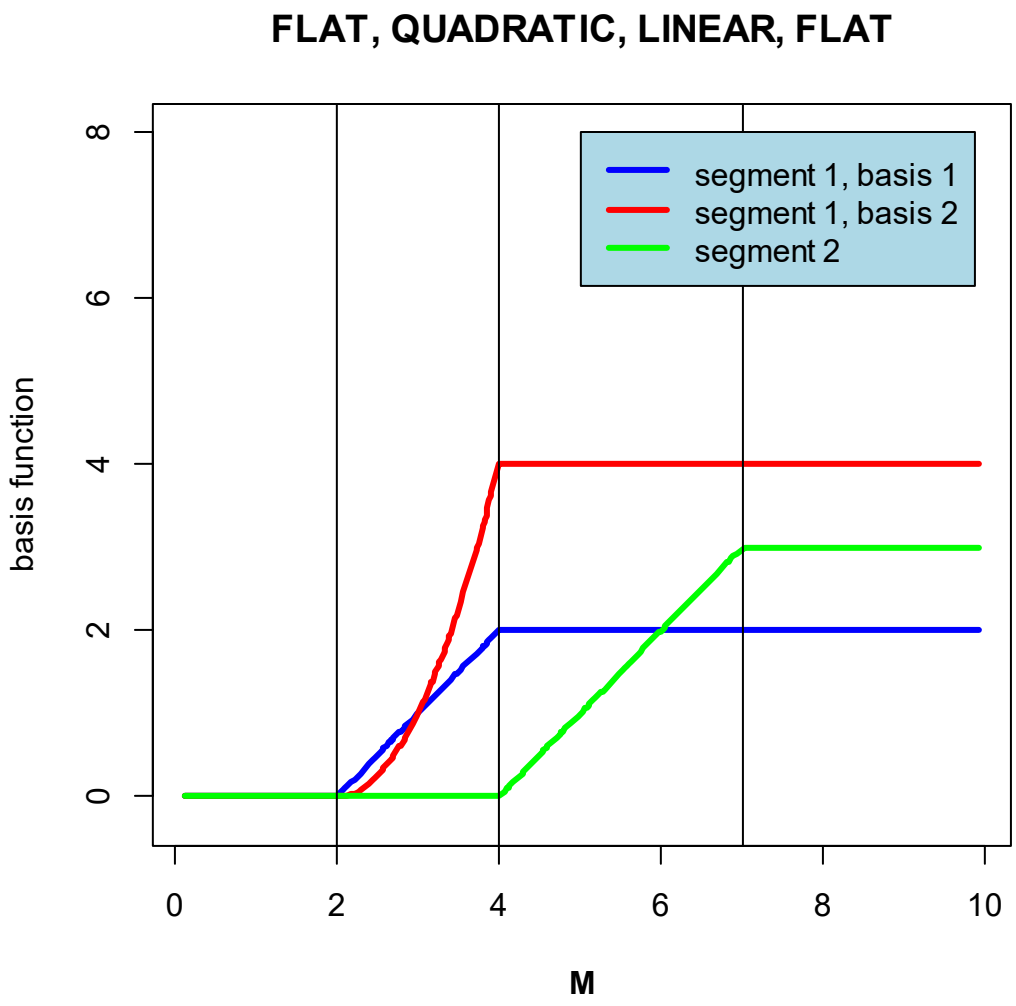
```
#-1.44767 -0.29439 -0.01816  0.34015  1.24359
#
#Coefficients:
#                    Estimate Std. Error t value Pr(>|t|)
#(Intercept)         -3.58901    0.06395 -56.119  < 2e-16 ***
#b1.1(M, c[1], c[2])  3.36741    0.21592  15.596  < 2e-16 ***
#b1.2(M, c[1], c[2]) -0.50595    0.10947  -4.622  5.7e-06 ***
#b2(M, c[2], c[3])    0.48282    0.04407  10.957  < 2e-16 ***
#b3(M, c[3])         -0.46991    0.05168  -9.093  < 2e-16 ***
#---

coef <- coefficients(Model)
#> coef
#        (Intercept) b1.1(M, c[1], c[2]) b1.2(M, c[1], c[2])   b2(M, c[2],
c[3])         b3(M, c[3])
#         -3.5890096           3.3674056          -0.5059541
0.4828228          -0.4699064

# Replot data
plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)
# Plot actual function
abline(v = c)
lines(Msorted,y(Msorted,c), lwd=3, col="red")


# Add predictions
yp <- coef[1]+coef[2]*b1.1(Msorted,c[1], c[2])+coef[3]*b1.2(Msorted,c[1],
      c[2]) +
      coef[4]*b2(Msorted,c[2],c[3]) +
      coef[5]*b3(Msorted,c[3])
lines(Msorted,yp, lwd=2, col="blue")
legend("bottomright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)", "fitted function"),
        lty=c(1,NA,1), lwd=c(3,NA,2), pch=c(NA,1,NA), col=c("red", "black",
"blue"), bg="lightblue")
```


FLAT, QUADRATIC, LINEAR, LINEAR
y
M
desired function
function with added noise
(data to be fit in regression)
fitted function

```
#-------------------------------------------------------------------------
----------------------
```
Title <- "FLAT, QUADRATIC, LINEAR, FLAT"
```
# No basis functions are needed for flat segments
# Define function

# Breakpoints:
c <- c(2, 4, 7)
y <- function(M,c){
  ifelse(M<c[1], -3.67,
    ifelse(M<c[2],1.13+1.4*(M-c[2])-0.5*(M-c[2])^2,
      ifelse(M<=c[3],1.13+0.5*(M-c[2]), 1.13+0.5*(c[3]-c[2]))))
 }


#Generate data
set.seed(1)
n <- 300
M <- runif(n, 0, 10)
Msorted <- sort(M)

# Add some noise:
yn <- y(M,c) + rnorm(n, sd = 0.5)

plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)
abline(v = c)
lines(Msorted, y(Msorted,c),lwd=3, col="red")
legend("bottomright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)"),
        lty=c(1,NA), lwd=c(3,NA), pch=c(NA,1), col=c("red", "black"),
bg="lightblue")
```

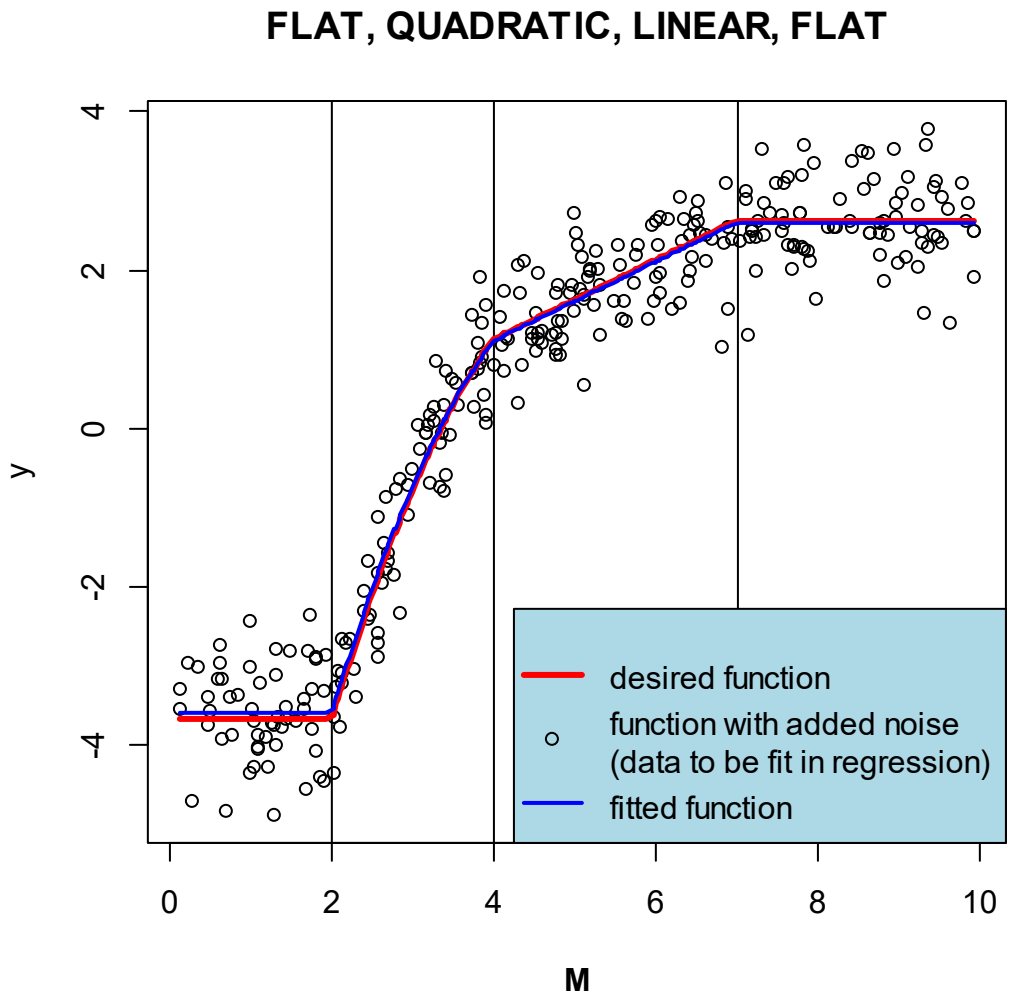



```
# Compute basis function for each region:
b1.1 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,x-L,R-L))}
b1.2 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,(x-L)^2,(R-L)^2))}
b2 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,x-L,R-L))}

plot(Msorted, b1.1(Msorted,c[1],c[2]), xlab=substitute(paste(bold("M"))),
ylab="basis function",
  ylim=c(-0.25,8.0), type="l", col = "blue", lwd=3, main=Title)
lines(Msorted, b1.2(Msorted,c[1],c[2]), col = "red", lwd=3)
lines(Msorted, b2(Msorted,c[2],c[3]), col = "green", lwd=3)
abline(v = c)
legend(5,8, legend=c("segment 1, basis 1","segment 1, basis 2", "segment 2"),
        lty=rep(1,3), lwd=rep(3,3), col=c("blue", "red", "green"),
bg="lightblue")
```




```
Model <- lm(yn ~ b1.1(M,c[1],c[2]) + b1.2(M,c[1],c[2]) + b2(M,c[2],c[3]) )
# Regression summary:
summary(Model)

#Call:
#lm(formula = yn ~ b2(M, c[1], c[2]) + b2.2(M, c[1], c[2]) + b3(M,
#    c[2], c[3]))
#
#Residuals:
#     Min       1Q   Median       3Q      Max
#-1.47883 -0.29232 -0.01076  0.33950  1.24418
#
#Coefficients:
#                   Estimate Std. Error t value Pr(>|t|)
#(Intercept)         -3.58959    0.06387 -56.198  < 2e-16 ***
#b2(M, c[1], c[2])    3.37824    0.21487  15.722  < 2e-16 ***
```

```
#b2.2(M, c[1], c[2]) -0.51440    0.10839  -4.746 3.24e-06 ***
#b3(M, c[2], c[3])    0.49841    0.03497  14.253  < 2e-16 ***
#---

coef <- coefficients(Model)
#> coef
#        (Intercept)   b2(M, c[1], c[2]) b2.2(M, c[1], c[2])   b3(M, c[2],
c[3])
#         -3.5895926           3.3782354          -0.5143986
0.4984074

# Replot data
plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)
# Plot actual function
abline(v = c)
lines(Msorted,y(Msorted,c), lwd=3, col="red")


# Add predictions
yp <- coef[1]+coef[2]*b1.1(Msorted,c[1], c[2])+coef[3]*b1.2(Msorted,c[1],
c[2]) +
      coef[4]*b2(Msorted,c[2],c[3])
lines(Msorted,yp, lwd=2, col="blue")
legend("bottomright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)", "fitted function"),
        lty=c(1,NA,1), lwd=c(3,NA,2), pch=c(NA,1,NA), col=c("red", "black",
"blue"), bg="lightblue")
```

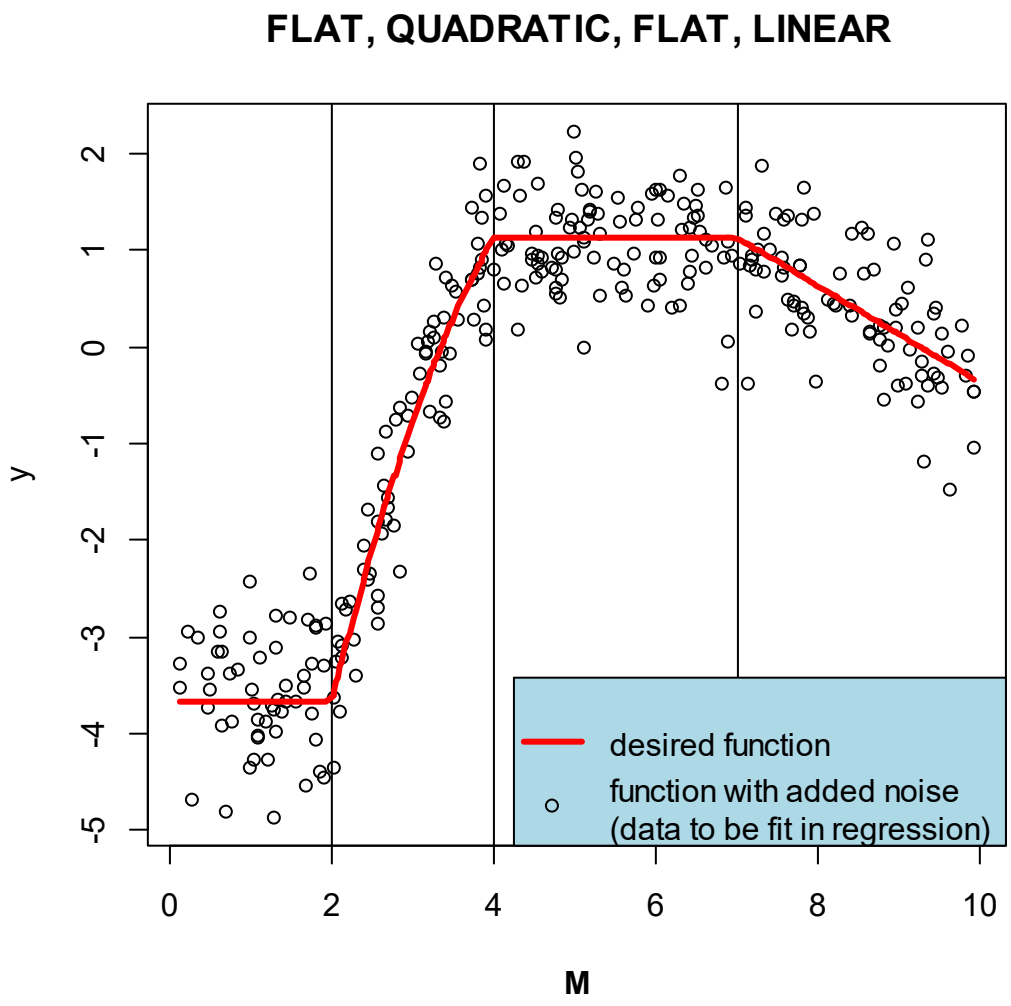

FLAT, QUADRATIC, LINEAR, FLAT
y
M
desired function
function with added noise
(data to be fit in regression)
fitted function

```
#-------------------------------------------------------------------------------
----------------------
```

Title <- "FLAT, QUADRATIC, FLAT, LINEAR"

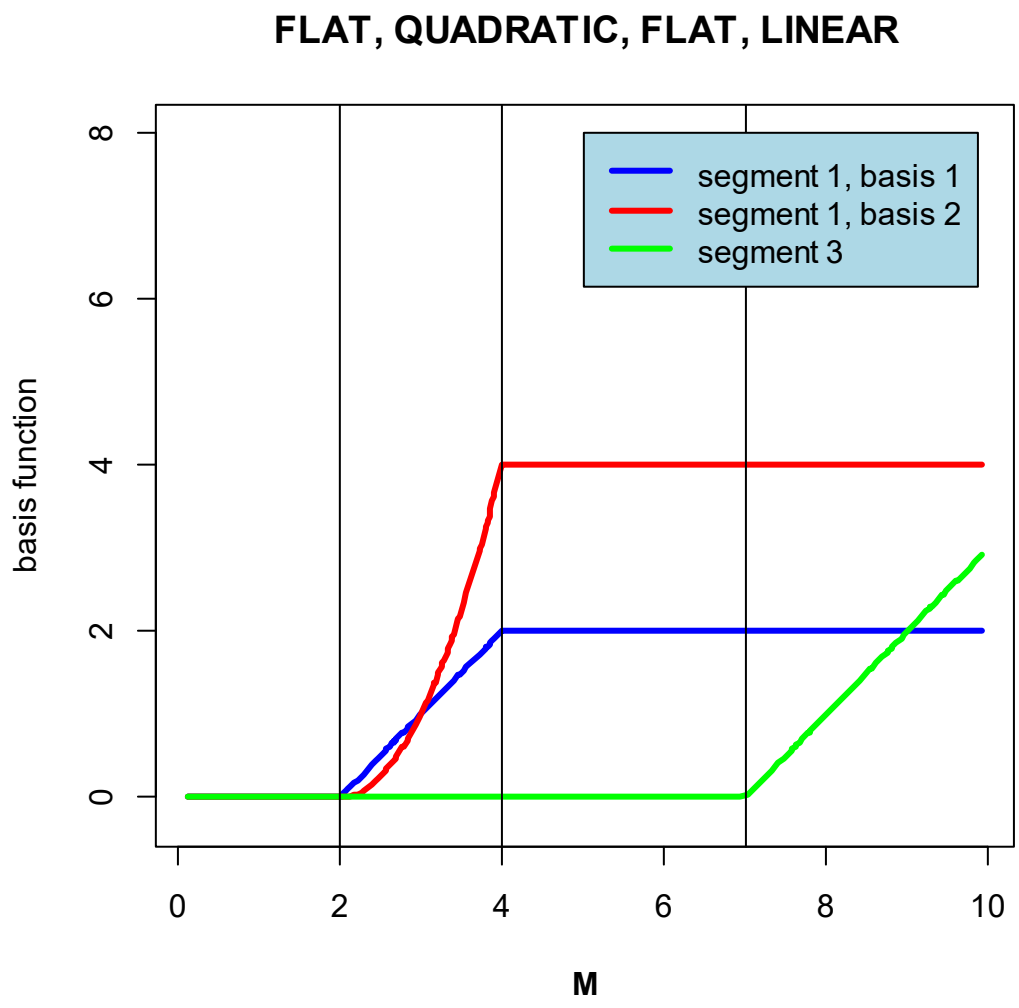
```
# No basis functions are needed for flat segments#

# Define function


# Breakpoints:

c <- c(2, 4, 7)

y <- function(M,c){

  ifelse(M<c[1], -3.67,

    ifelse(M<c[2],1.13+1.4*(M-c[2])-0.5*(M-c[2])^2,

      ifelse(M<=c[3],1.13, 1.13-0.5*(M-c[3]))))

 }



#Generate data

set.seed(1)

n <- 300

M <- runif(n, 0, 10)

Msorted <- sort(M)


# Add some noise:

yn <- y(M,c) + rnorm(n, sd = 0.5)


plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)

abline(v = c)

lines(Msorted, y(Msorted,c),lwd=3, col="red")
```

```
legend("bottomright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)"),

        lty=c(1,NA), lwd=c(3,NA), pch=c(NA,1), col=c("red", "black"),
bg="lightblue")
```



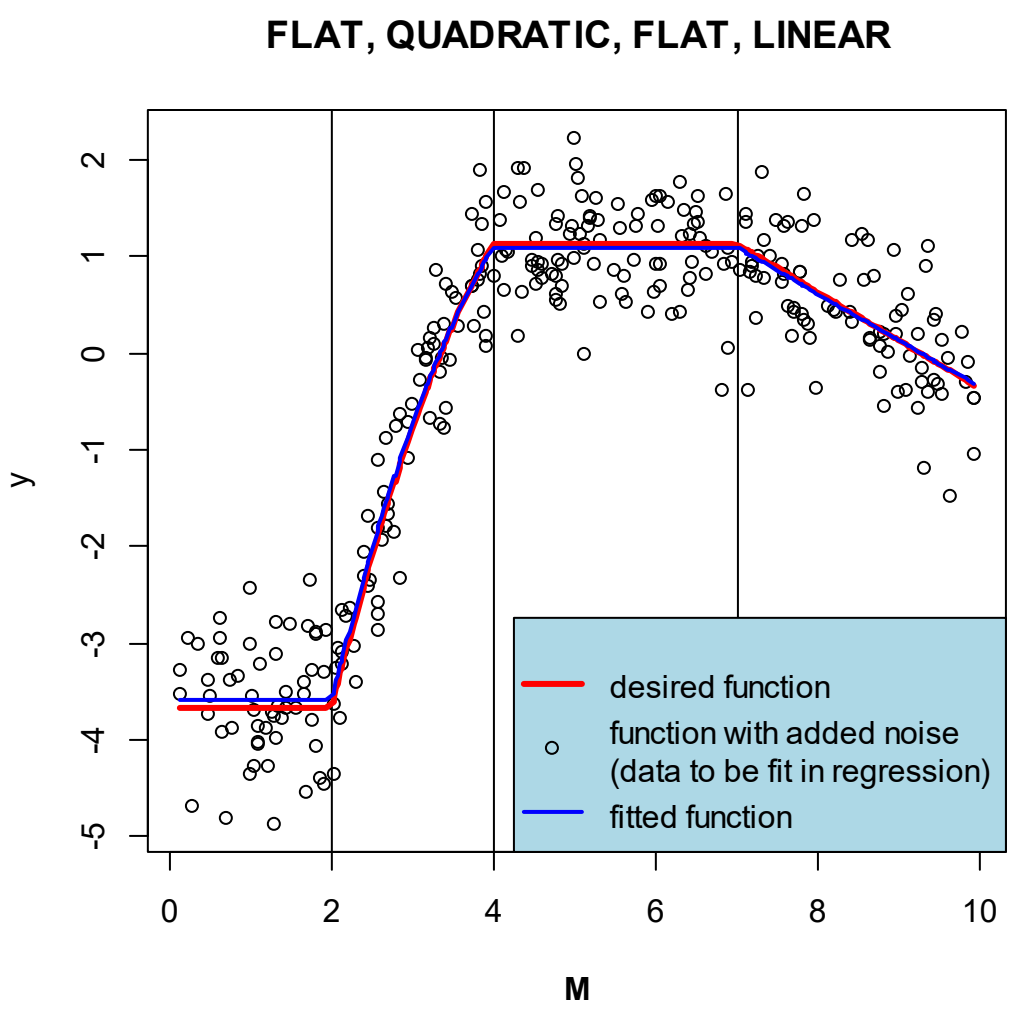

```
# Compute basis function for each region:

b1.1 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,x-L,R-L))}

b1.2 <- function(x,L,R){ifelse(x<L,0,ifelse(x<R,(x-L)^2,(R-L)^2))}

b3 <- function(x,L){ifelse(x<L,0,x-L)}
```

```
plot(Msorted, b1.1(Msorted,c[1],c[2]), xlab=substitute(paste(bold("M"))),
ylab="basis function",

  ylim=c(-0.25,8.0), type="l", col = "blue", lwd=3, main=Title)

lines(Msorted, b1.2(Msorted,c[1],c[2]), col = "red", lwd=3)

lines(Msorted, b3(Msorted,c[3]), col = "green", lwd=3)

abline(v = c)

legend(5,8, legend=c("segment 1, basis 1","segment 1, basis 2", "segment 3"),

        lty=rep(1,3), lwd=rep(3,3), col=c("blue", "red", "green"),
bg="lightblue")
```



```
Model <- lm(yn ~ b1.1(M,c[1],c[2]) + b1.2(M,c[1],c[2]) + b3(M,c[3]) )
# Regression summary:
summary(Model)


#Call:
#lm(formula = yn ~ b1.1(M, c[1], c[2]) + b1.2(M, c[1], c[2]) +
#    b3(M, c[3]))
#
#Residuals:
#     Min       1Q   Median       3Q      Max
#-1.46848 -0.29279 -0.01513  0.34001  1.24486
#
#Coefficients:
#                    Estimate Std. Error t value Pr(>|t|)
#(Intercept)         -3.59027    0.06378 -56.292  < 2e-16 ***
#b1.1(M, c[1], c[2])  3.39088    0.20705  16.377  < 2e-16 ***
#b1.2(M, c[1], c[2]) -0.52426    0.09875  -5.309 2.17e-07 ***
#b3(M, c[3])         -0.48214    0.04100 -11.761  < 2e-16 ***
#---
#Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
#Residual standard error: 0.5123 on 296 degrees of freedom
#Multiple R-squared:  0.9271,    Adjusted R-squared:  0.9264
#F-statistic:  1256 on 3 and 296 DF,  p-value: < 2.2e-16#Call:


coef <- coefficients(Model)
> coef
```

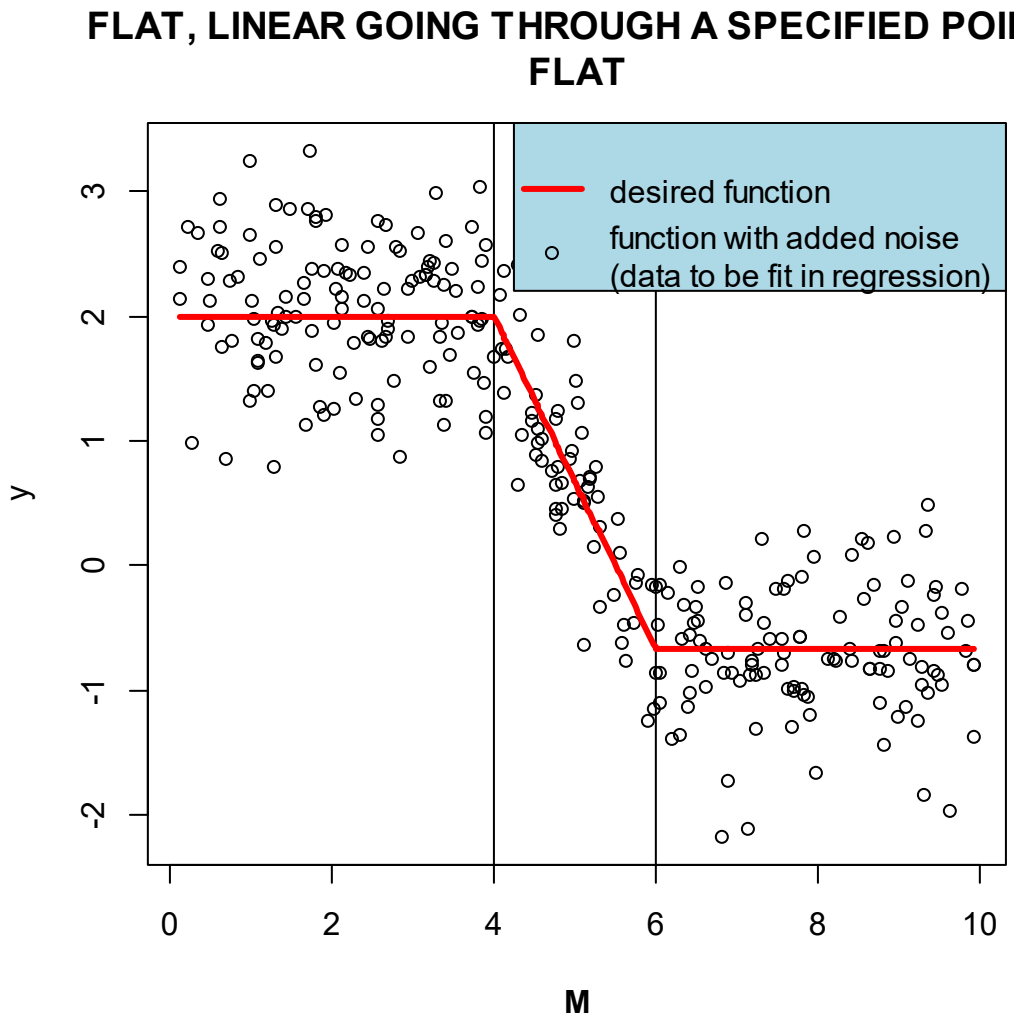
```
        (Intercept) b1.1(M, c[1], c[2]) b1.2(M, c[1], c[2])         b3(M,
c[3])

         -3.5902734           3.3908799          -0.5242581          -
0.4821406



# Replot data

plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main=Title)

# Plot actual function

abline(v = c)

lines(Msorted,y(Msorted,c), lwd=3, col="red")




# Add predictions

yp <- coef[1]+coef[2]*b1.1(Msorted,c[1], c[2])+coef[3]*b1.2(Msorted,c[1],
c[2]) +

      coef[4]*b3(Msorted,c[3])

lines(Msorted,yp, lwd=2, col="blue")

legend("bottomright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)", "fitted function"),

        lty=c(1,NA,1), lwd=c(3,NA,2), pch=c(NA,1,NA), col=c("red", "black",
"blue"), bg="lightblue")
```


FLAT, QUADRATIC, FLAT, LINEAR
y
M
desired function
function with added noise
(data to be fit in regression)
fitted function

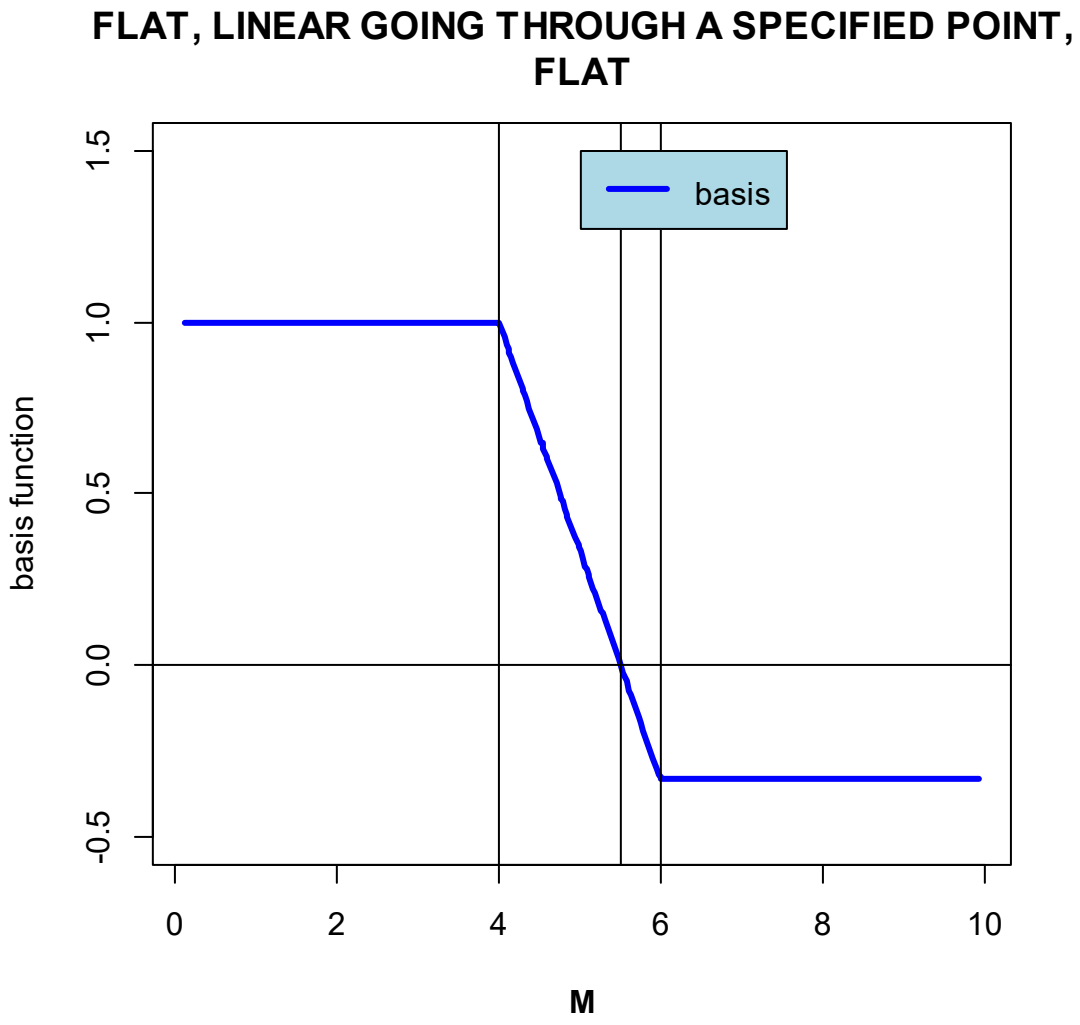
FLAT, LINE CROSSING 0.0 AT XC, FLAT

How should a model that is flat to c[1], forced to cross 0.0 at x = xc, and then is flat beyond c[2] be specified? Thinking about it, there is only one basis function, even though it has a break at x=c[1]. The reason is that the slope of the line between c[1] and c[2] is determined by the value of constant portion for x < c[1] and the condition that the line crosses the zero line at x=xc. Therefore, the slope is NOT a regression parameter. There is only one regression parameter.

```
#FLAT, LINEAR GOING THROUGH A SPECIFIED POINT, FLAT

# Define function

# Breakpoints:
c <- c(4, 6)
cz <- 5.5
slope <- (2.0-0.0)/(cz-c[1])
y <- function(x,c){
  ifelse(x<c[1], 2.0,
    ifelse(x<c[2],2.0 - slope*(x-c[1]), 2.0 - slope*(c[2]-c[1]) ))
 }


#Generate data
set.seed(1)
n <- 300
M <- runif(n, 0, 10)
Msorted <- sort(M)

# Add some noise:
yn <- y(M,c) + rnorm(n, sd = 0.5)

plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main="FLAT, LINEAR GOING THROUGH A SPECIFIED POINT,\nFLAT")
abline(v = c)
lines(Msorted, y(Msorted,c),lwd=3, col="red")
legend("topright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)"),
        lty=c(1,NA), lwd=c(3,NA), pch=c(NA,1), col=c("red", "black"),
bg="lightblue")
```

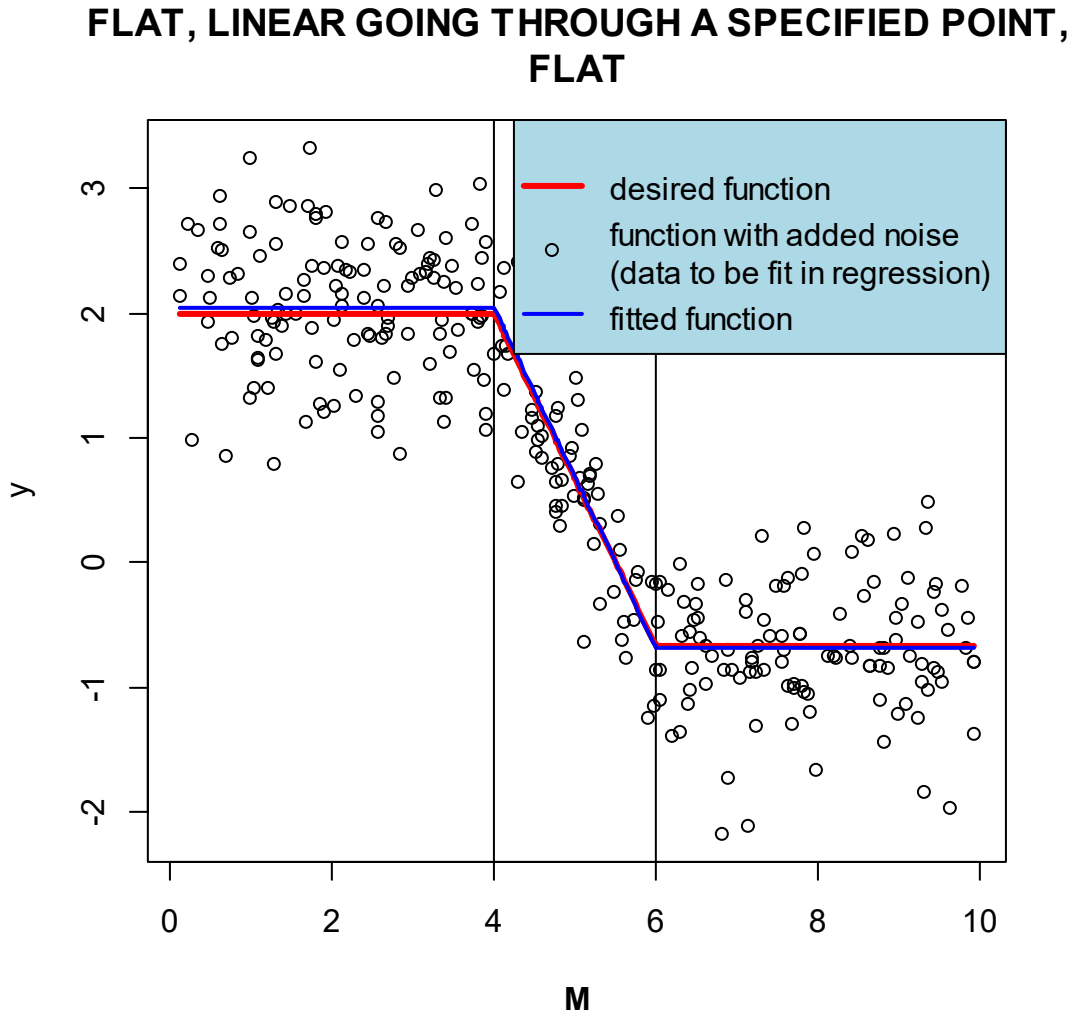



```
b1 <- function(x,L,xc,R){ifelse(x<L,1,ifelse(x<R,1-(x-L)/(xc-L),1-(R-L)/(xc-
L)))}

plot(Msorted, b1(Msorted,c[1],cz,c[2]), xlab=substitute(paste(bold("M"))),
ylab="basis function",
  ylim=c(-0.5,1.5), type="l", col = "blue", lwd=3, main="FLAT, LINEAR GOING
THROUGH A SPECIFIED POINT,\nFLAT")
abline(h=0)
abline(v = c(c[1],cz,c[2]))
legend(5,1.5, legend=c("basis"),
        lty=rep(1,1), lwd=rep(3,1), col=c("blue"), bg="lightblue")
```




```
Model <- lm(yn ~ -1 + b1(M,c[1],cz,c[2]) )
# Regression summary:
summary(Model)

#Call:
#lm(formula = yn ~ -1 + b1(M, c[1], cz, c[2]))
#
#Residuals:
#     Min       1Q   Median       3Q      Max
#-1.48759 -0.30188 -0.03717  0.33624  1.27527
#
#Coefficients:
#                      Estimate Std. Error t value Pr(>|t|)
#b1(M, c[1], cz, c[2])   2.0493     0.0411   49.86   <2e-16 ***
#---
```

```
#Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
#Residual standard error: 0.5107 on 299 degrees of freedom
#Multiple R-squared:  0.8927,    Adjusted R-squared:  0.8923
#F-statistic:  2486 on 1 and 299 DF,  p-value: < 2.2e-16

coef <- coefficients(Model)
coef
#b1(M, c[1], cz, c[2])
#              2.04931

# Replot data
plot(M, yn,  xlab=substitute(paste(bold("M"))), ylab="y", col="black",
main="FLAT, LINEAR GOING THROUGH A SPECIFIED POINT,\nFLAT")
# Plot actual function
abline(v = c)
lines(Msorted,y(Msorted,c), lwd=3, col="red")


# Add predictions
yp <- coef[1]*b1(Msorted,c[1],cz,c[2])
lines(Msorted,yp, lwd=2, col="blue")
legend("topright", legend=c("desired function", "function with added
noise\n(data to be fit in regression)", "fitted function"),
        lty=c(1,NA,1), lwd=c(3,NA,2), pch=c(NA,1,NA), col=c("red", "black",
"blue"), bg="lightblue")
```


FLAT, LINEAR GOING THROUGH A SPECIFIED POINT, FLAT
desired function
function with added noise (data to be fit in regression)
fitted function
y
M
3
2
1
0
-1
-2
0
2
4
6
8
10