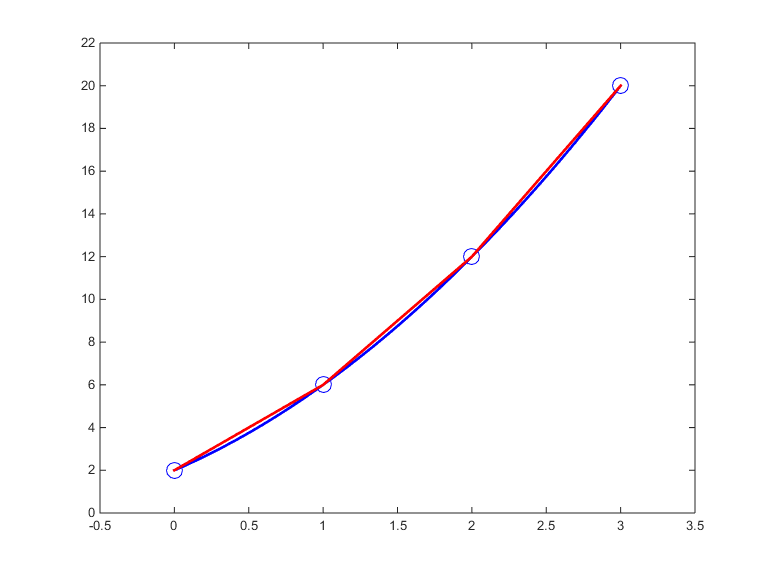
**PC5a - Interpolation**

**1.** In this question, you will approximate a quadratic equation by piecewise linear interpolation. The graph below shows the function (blue curve) and its linear approximation (red line). The linear approximation is based on the points (0,2), (1,6), (2,12) and (3,20).

Determine the difference (error) between the function and its linear approximation for the points 1/2, 3/2 and 5/2.

You can use

x <- 0:3

y <- x^2+3\*x+2;

The linear equation based on the points can be determined by

> library(pracma)

> p <- polyfit(x[1:2],y[1:2],1); p

[1] 4 2

This represents the equation For *x* is 1/2, the equation equals

>> polyval(p,1/2)

[1] 4

Use the statements above and a for loop to determine the errors for the points 1/2, 3/2 and 5/2.

Example output:

x=0.500 y(true)= 3.750 y(approximation)= 4.000 error=-0.250

x=1.500 y(true)= 8.750 y(approximation)= 9.000 error=-0.250

x=2.500 y(true)=15.750 y(approximation)=16.000 error=-0.250

**2.** Determine using Newtons forward differences the quadratic function that goes through the points , and

Write an R-script that displays the following table:

x f df d2f

=================================================

0.000000 0.000000 1.000000 -2.000000

1.570796 1.000000 -1.000000

3.141593 0.000000

Subsequently calculate by hand: with Check using polyfit.

**3.** Determine by hand the 3rd order Lagrange polynomial that goes through the points , , and You should use the formula:



Again, check your answer using polyfit.

**4.** Plot de spline that goes through the points , , , and for the interval in steps of Use the R function pracma::cubicspline.

**5.** Determine the value *a* and *b* in using the least squares method and the following data set:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x* | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| *y* | 3.16 | 3.01 | 2.73 | 2.47 | 2.13 | 1.82 | 1.52 | 1.21 | 0.76 | 0.43 | 0.03 |

First, clean the R global environment and read the data (use YX.txt on **Canvas**) :

rm(list = ls())

data <- read.table(…,header=TRUE)

Then write an R-function:

SSR <- function(coef,y,x){

}

The input argument coef is a 1×2 vector that contains the coefficients. So, coef(1) contains the parameter *a* and coef(2) the parameter *b*. The output of the function should be the sum of squared errors, i.e.

The parameters *a* and *b* can be estimated by

results <- optim(c(0,0),SSR,gr=NULL,y,x,method = c("BFGS"))

What are optimal values of *a* and *b* according to least squares?

What does the vector c(0,0) represent?

**6.** [variant on 4] Determine the fitted spline values that goes through the points , , , and for the interval in steps of (the step size is different from question 4).

**a.** If is the true function, determine the sum of squared errors (of the spline approximation) based on the points seq(-pi,pi,pi/10).

**b.** Show the coefficients of the 4 splines.

**c.** Verify that the first and second-order derivate of the spline from the left side matches the first and second-order derivate from the right side in the point (0,0).

setwd("D:/UvA/Year 1/Block 5/Programming and Numerical Analysis/Week 5")

rm(list = ls())

#1

x <- 0:3

y <- x^2+3\*x+2

library("pracma")

p <- polyfit(x[1:2], y[1:2], 1); p

polyval(p, 1/2)

fy <- function(x) {

x^2+3\*x+2

}

error <- NULL

for (i in 0:2) {

p <- polyfit(x[(i+1):(i+2)], y[(i+1):(i+2)], 1)

pvalue <- polyval(p, i+0.5)

error[i+1] <- fy(i+0.5) - pvalue

cat(sprintf("x=%.3f y(true)=%6.3f y(approx)=%6.3f error=%6.3f", i+0.5, fy(i+0.5), pvalue, error[i+1])); cat("\n")

}

#2

x <- c(0, pi/2, pi)

f <- c(0, 1, 0)

df <- c(f[2]-f[1], f[3]-f[2])

d2f <- c(df[2]-df[1])

#OR

df <- diff(f)

d2f <- diff(df)

cat("x f df d2f", "\n")

cat("==========================", "\n")

cat(sprintf("%9.6f %8.6f %8.6f %8.6f", x[1], f[1], df[1], d2f[1]), "\n")

cat(sprintf("%9.6f %8.6f %8.6f", x[2], f[2], df[2]), "\n")

cat(sprintf("%8.6f %8.6f", x[3], f[3]), "\n")

polyfit(x[1:3], y[1:3], 2)

#3

x <- c(-pi/2, 0, pi/2, pi)

y <- c(-1,0,1,0)

p <- polyfit(x, y, 3)

y1 <- polyval(p,x)

plot(x,y)

lines(x, y1, col = "red")

#4

?cubicspline

x <- c(-pi, -pi/2, 0, pi/2, pi)

y <- c(0, -1, 0, 1, 0)

xs <- seq(-pi, pi, by = pi/20)

pp <- cubicspline(x, y)

y1 <- ppval(pp, xs)

plot(x, y, ylim=c(-1.5, 1.5), xlim = c(-4,4))

lines(y1)

plot(x, y, ylim=c(-1.5, 1.5), xlim = c(-4,4))

lines(spline(x,y), col="blue")

x<-c(-pi, -pi/2, 0, pi/2, pi)

y<-c(0, -1, 0, 1, 0)

plot(cubicspline(x, y, xi = NULL, endp2nd = FALSE, der = c(0, 0)))

a <- cubicspline(x, y, xi = NULL, endp2nd = FALSE, der = c(0, 0))

y1 <- ppval(a, xs)

y1

#5

data <- read.table(file = "YX.txt",header=TRUE)

View(data)

SSR <- function(coef, y, x) {

}