Zusammenfassung der Experimente und Auswertung

Charakterisierung der Bauteile

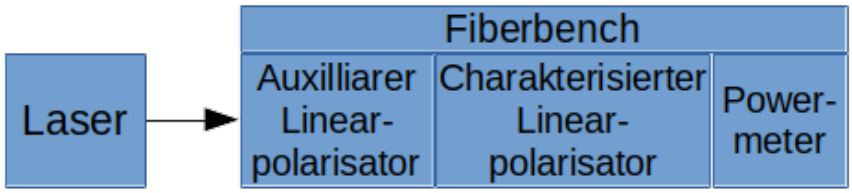
Linearpolarisatoren

Experiment

Es werden zwei Experimente durchgeführt:

1. Nullpunktbestimmung: Welche Position des Linearpolarisators ist parallel/orthogonal zum Laser?
2. Transmissionsverhalten: Ist der Linearpolarisator unterschiedlich durchlässig für einen parallel polarisierten Laser, wenn sowohl der Laser als auch der Polarisator im gleichen Maß gedreht werden?

Nullpunktsbestimmung von Linearpolarisatoren:

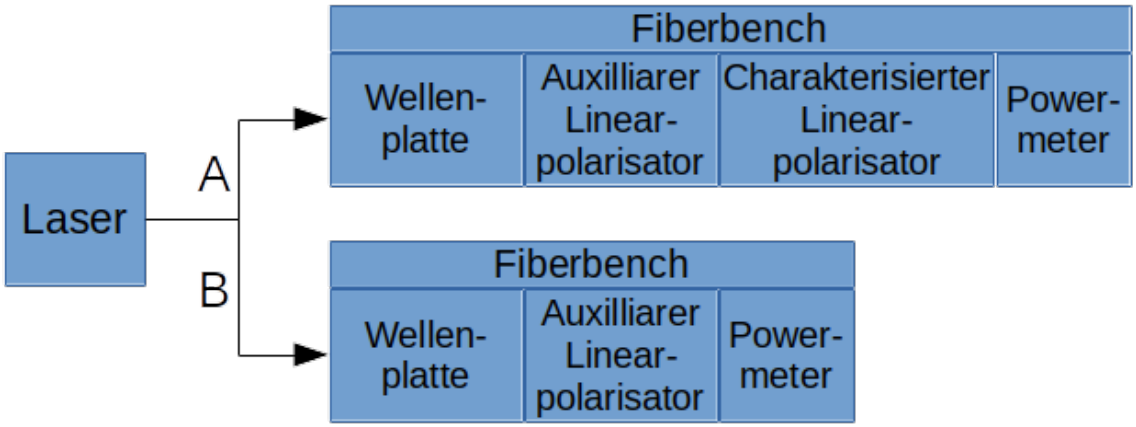


Aufbau Nullpunktsbestimmung

Durchführung:

- Auxilliar Polarisator parallel zum Laser ausrichten
- Leistung für verschiedene Rotationen des zu charakterisierenden Polarisators

Transmissionsverhalten von Linearpolarisatoren (Variante eine Fiberbench):

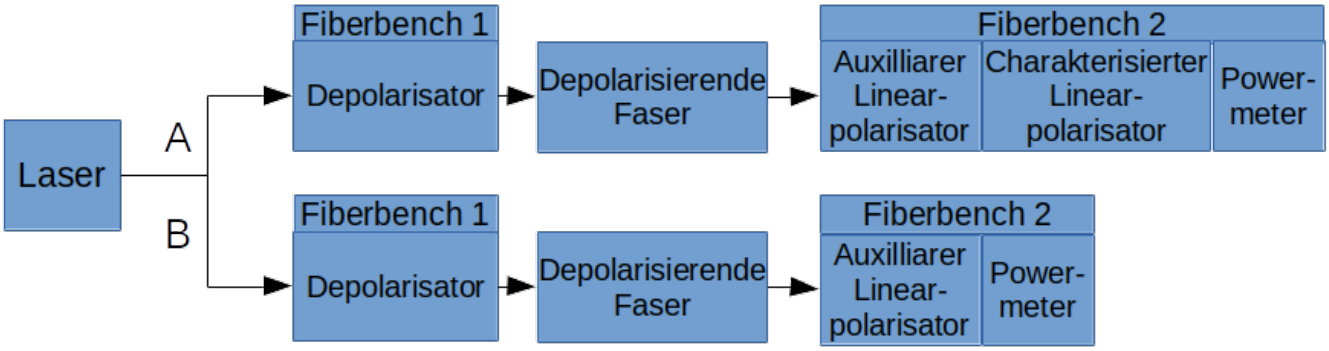


Aufbau Transmissionsverhalten von Linearpolarisatoren (A: Egtl. Messung, B: Normierung)

Durchführung:

- Verwendet für Polarisatoren, die nur eine Position in der Fiberbench einnehmen
- Für beliebige Positionen der Wellenplatte werden folgende Schritte wiederholt
- Der auxilliare Polarisator wird so gedreht, dass die Transmission maximal ist
- Der zu charaktarisierende Polarisator wird so gedreht, dass die Transmission maximal ist
- Leistung wird vor und hinter den zu charakterisierenden Polarisator gemessen

Transmissionsverhalten von Linearpolarisatoren (Variante zwei Fiberbenches):



Aufbau Transmissionsverhalten von Linearpolarisatoren (A: Egtl. Messung, B: Normierung)

- Depolarisator und depolarisierende Faser ersetzten die Wellenplatte im Aufbau mit einer Fiberbench
- Wellenplatte kann nicht verwendet werden, weil Linearpolarisatoren P3 und P4 zu lang sind, um alle Polarisatoren und die Wellenplatte in eine Fiberbench zu stellen
- Wellenplatte wichtig, um sicher zu stellen, dass die messbare Leistung so groß wie möglich ist

Durchführung:

- Für beliebige Positionen des zu charakterisierenden Linearpolarisators wird der Auxilliarpolarisator auf maximale Transmission gedreht
- Die Leistung wird vor und hinter dem zu charakterisierenden Polarisator gemessen

Auswertung

Nullpunktbstimmung

- Herunterladen von Messdaten
- Normalisieren der Daten $P_{measured}$ mit der Leistung gemessen ohne Linearpolarisator P_0: $P = \frac{P_{measured}}{P_0} \cdot 100\%$

```
#
# LINEARPOLARISATOR P1
#
# Extract Data from eLabFTW
P1.maxmin.metadata <- GET.elabftw.byselector(25, node.selector = "#meta-data")[[1]]
P1.maxmin.data <- GET.elabftw.byselector(25, header = T)[[1]]

# Normalise data
P1.maxmin.data$Y1 <- P1.maxmin.data$Y1 / P1.maxmin.metadata[3,2] *100

#
# LINEARPOLARISATOR P2
#
# Extract Data from eLabFTW
P2.maxmin.metadata <- GET.elabftw.byselector(26, node.selector = "#meta-data")[[1]]
P2.maxmin.data <- GET.elabftw.byselector(26, header = T)[[1]]

# Normalise data
P2.maxmin.data$Y1 <- P2.maxmin.data$Y1 / P2.maxmin.metadata[3,2] *100
```

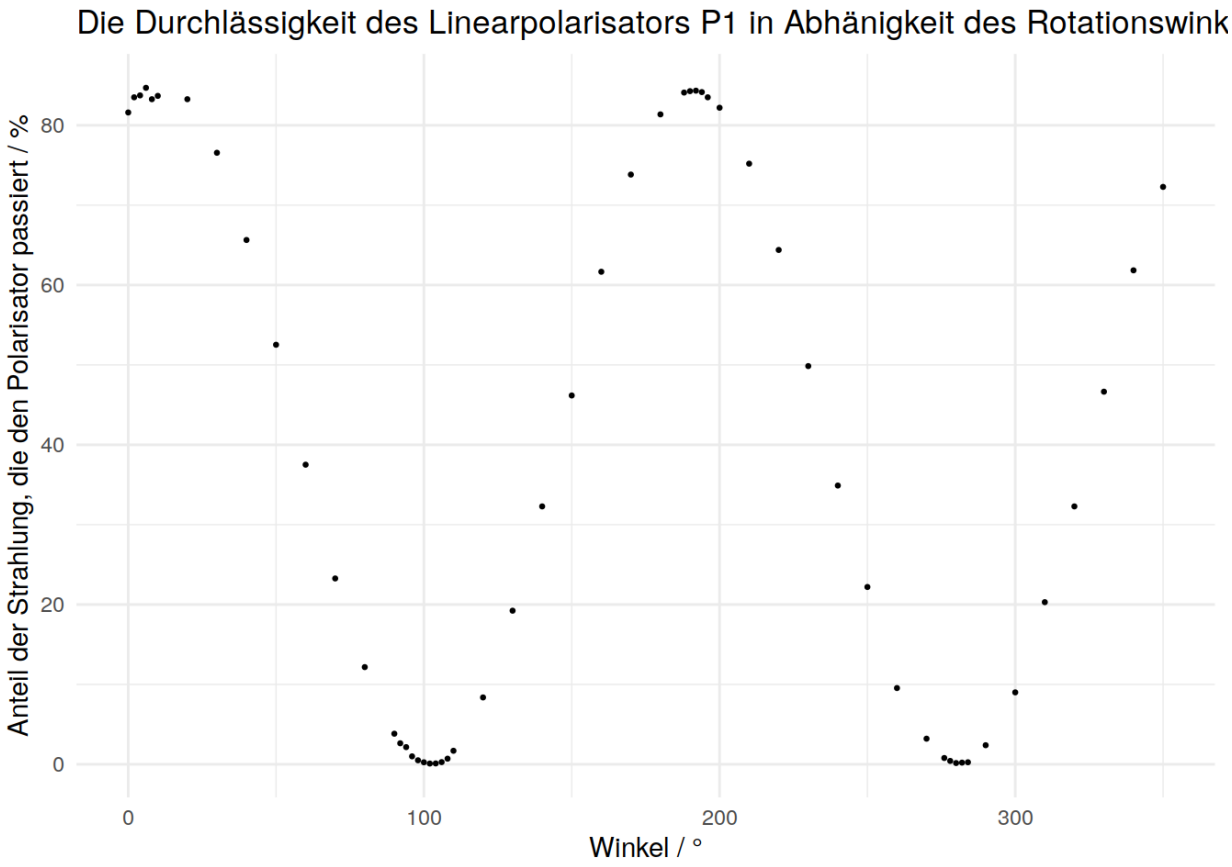
Winkelabhängige Transmission

- Herunterladen der Messdaten
- Ggf. .csv Dateien auslesen und den Mittelwert (unter Vernachlässigung der ersten und letzten 10%-Quantille) der gemessenen Leistungen nehmen (Funktionen qmean parseTable.elabftw erlediegen das)
- Normalisieren der Leistung gemessen hinter dem Polarisator P_1 mit der Leistung gemessen vor dem Polarisator P_0 : P=\frac{P_1}{P_0} \cdot 100\%

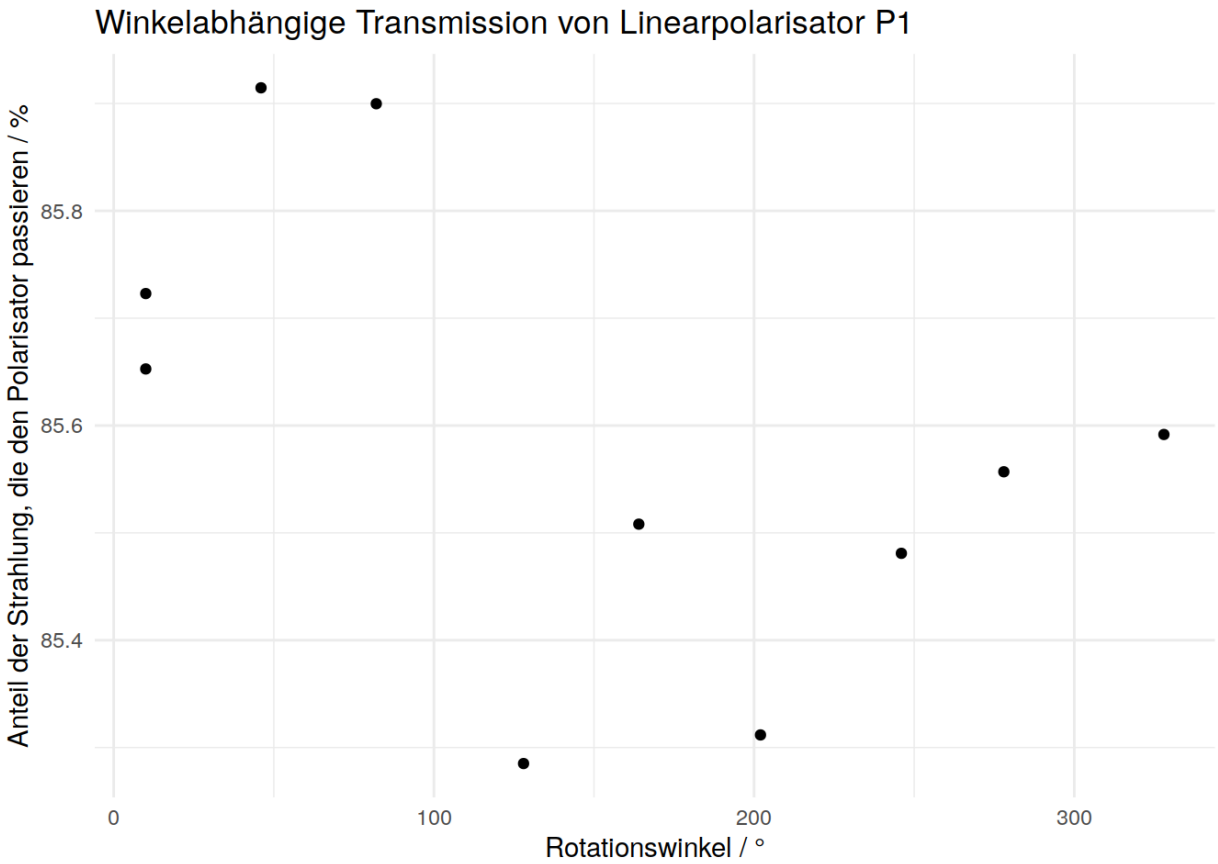
```
#
# LINEARPOLARISATOR P1
#
# Extract data from eLabFTW
P1.transmission.data <- GET.elabftw.byselector(29, header = T)[[1]]

# Normalise data
P1.transmission.data$Y4 <- P1.transmission.data$Y4/P1.transmission.data$Y2 *100

#
# LINEARPOLARISATOR P2
#
# Extract data from eLabFTW
P2.transmission.data <- GET.elabftw.byselector(30, header = T)[[1]]

# Normalise data
P2.transmission.data$Y4 <- P2.transmission.data$Y4/P2.transmission.data$Y2 *100

#
# LINEARPOLARISATOR P3
#
# Extract data from elabFTW
P3.absorbance <- GET.elabftw.bycaption(78, header=T, outputHTTP=T) %>%
                  parseTable.elabftw(., func=function(x) qmean(x[,4], 0.8, na.rm=T, inf.rm
=T),
                                     header=T, skip=14, sep=";") %>%
                  .[[1]]
colnames(P3.absorbance) <- c("P3", "P4", "background", "measured")

# Normalise data
P3.absorbance$transmittance <- P3.absorbance$measured / P3.absorbance$background
```

Linearpolarisator P1

```
# Plot Zero Point
ggplot(data = P1.maxmin.data, mapping = aes(x = X, y = Y1)) +
  geom_point(size = 0.5) +
  theme_minimal() +
  labs(title = "Die Durchlässigkeit des Linearpolarisators P1 in Abhänigkeit des Rotations
winkels",
       x = "Winkel / °",
       y = "Anteil der Strahlung, die den Polarisator passiert / %")
```




```
# Plot Transmissionsverhalten
ggplot(data = P1.transmission.data, mapping = aes(x = Y3, y = Y4)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Winkelabhängige Transmission von Linearpolarisator P1",
       x = "Rotationswinkel / °",
       y = "Anteil der Strahlung, die den Polarisator passieren / %")
```

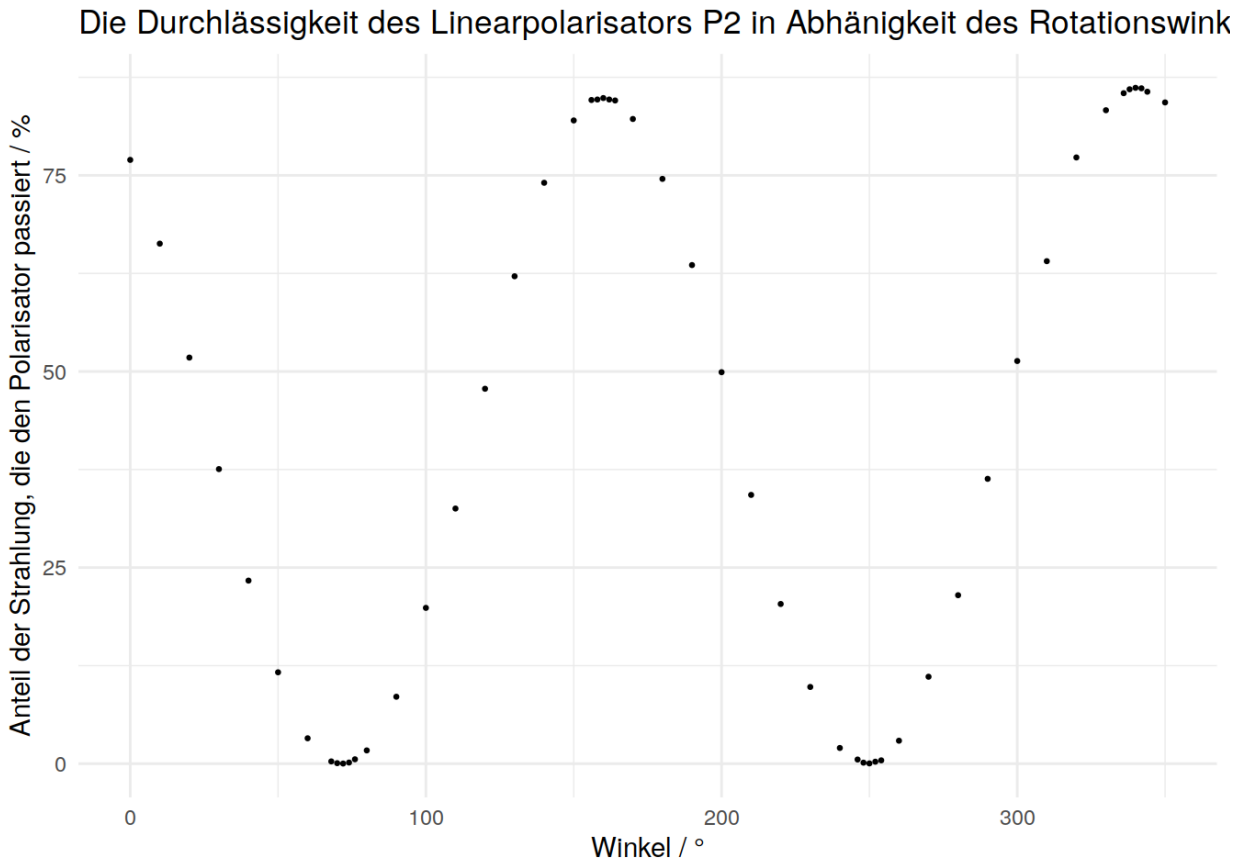


Fazit:

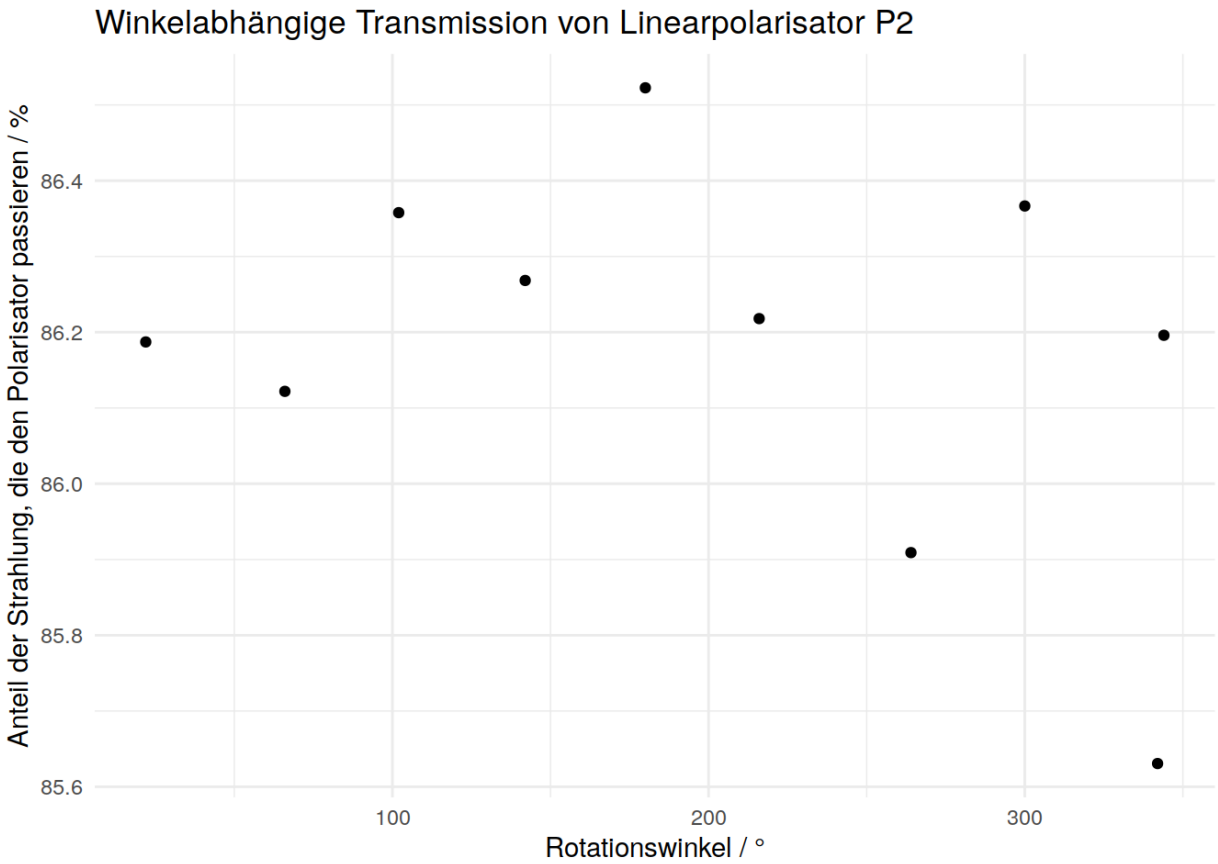
- Rotation parallel zu Laser: 6°, 192°
- Rotation orthogonal zu Laser: 102°, 280°
- Transmissionsverhalten schwankt wenig und scheint keinem Muster zu folgen (+/- 0.5%?)
- Transmittiert ca. 85.6%
- Winkelabhängigkeit der Transmission wie erwartet

Linearpolarisator P2

```
# Plot Zero Point
ggplot(data = P2.maxmin.data, mapping = aes(x = X, y = Y1)) +
  geom_point(size = 0.5) +
  theme_minimal() +
  labs(title = "Die Durchlässigkeit des Linearpolarisators P2 in Abhänigkeit des Rotations
winkels",
       x = "Winkel / °",
       y = "Anteil der Strahlung, die den Polarisator passiert / %")
```




```
# Plot Transmissionsverhalten
ggplot(data = P2.transmission.data, mapping = aes(x = Y3, y = Y4)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Winkelabhängige Transmission von Linearpolarisator P2",
       x = "Rotationswinkel / °",
       y = "Anteil der Strahlung, die den Polarisator passieren / %")
```

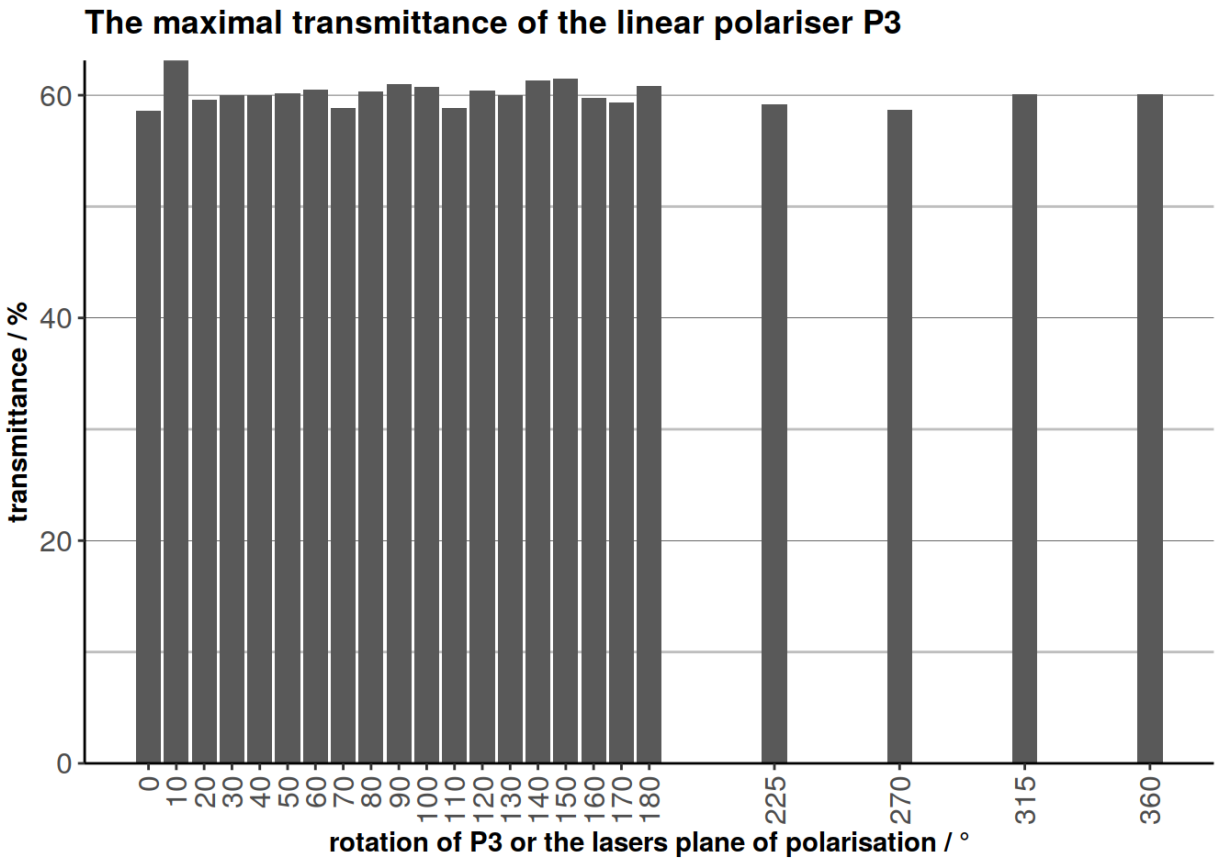


Fazit:

- Rotation parallel zu Laser: 158°, 340°
- Rotation orthogonal zu Laser: 72°, 250°
- Transmissionsverhalten schwankt wenig und scheint keinem Muster zu folgen (+/- 0.5%?)
- Transmittiert ca. 86.2%
- Winkelabhängigkeit der Transmission wie erwartet

Linearpolarisator P3

```
# Plot transmission behaviour
# Does the transmittance of the linear polariser change when rotating the polariser and th
e laser in the same manner?
ggplot(data = P3.absorbance,
       mapping = aes(x = P3, y = transmittance*100) ) +
  geom_bar(stat="identity") +
  theme_classic() +
  theme( axis.text = element_text(size=12),
         axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
         panel.grid.major.y = element_line("black", size = 0.1),
         panel.grid.minor.y = element_line("grey", size = 0.5) ) +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_continuous(breaks = P3.absorbance$P3) +
  labs(title = expression(bold("The maximal transmittance of the linear polariser P3")),
       x = expression(bold("rotation of P3 or the lasers plane of polarisation / °")),
       y = expression(bold("transmittance / %"))
       )
```



Fazit:

- Transmission schwankt wenig (+/- 1% bis zu 2.5%, stärker als P1/P2) und scheint unabhängig von der Laserpolarisation zu sein
- Transmittiert ca. 60% (weniger als P1/P2)
- Winkelabhängigkeit der Transmission wie erwartet

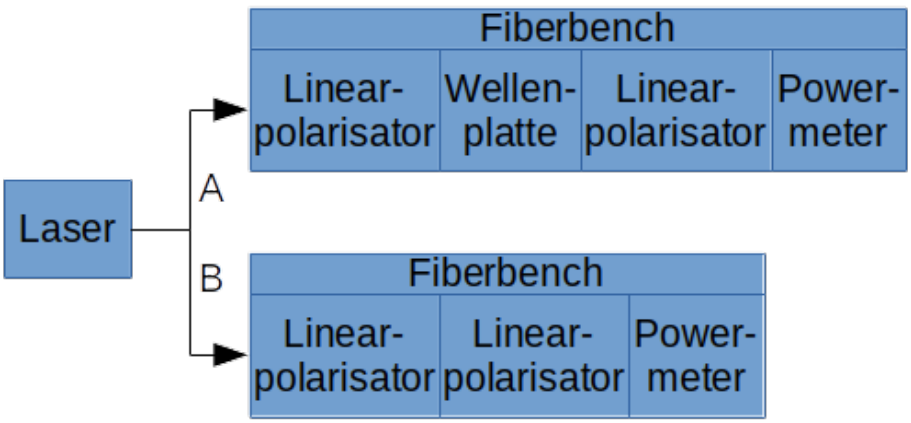
Wellenplatten

Experiment

Es werden zwei Experimente durchgeführt:

1. Nullpunktsbestimmung: Wie muss die Wellenplatte orientiert sein, um die Laserpolarisation um 90° zu drehen?
2. Transmissionsverhalten: Absorbiert die Wellenplatte unterschiedlich für verschiedene Orientierungen der Wellenplatte?

Nullpunktbestimmung von Wellenplatten:

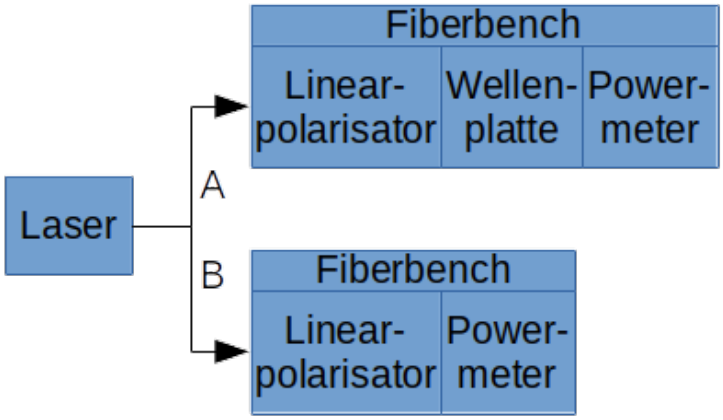


Aufbau Nullpunktbestimmung von Wellenplatten (A: Egtl. Messung, B: Normierung)

Durchführung:

- Der erste Linearpolarisator wird parallel zum Laser ausgerichtet
- Der zweite Linearpolarisator wird orthogonal zum Laser ausgerichtet
- Für verschiedene Positionen der Wellenplatte wird die Leistung sowohl mit als auch ohne Wellenplatte gemessen

Winkelabhängiges Transmissionsverhalten von Wellenplatten:



Aufbau Transmissionsverhalten von Wellenplatten (A: Egtl. Messung, B: Normierung)

Durchführung:

- Der Linearpolarisator wird parallel zum Laser ausgerichtet
- Für verschiedene Positionen der Wellenplatte wird die Leistung vor und hinter der Wellenplatte gemessen

Auswertung

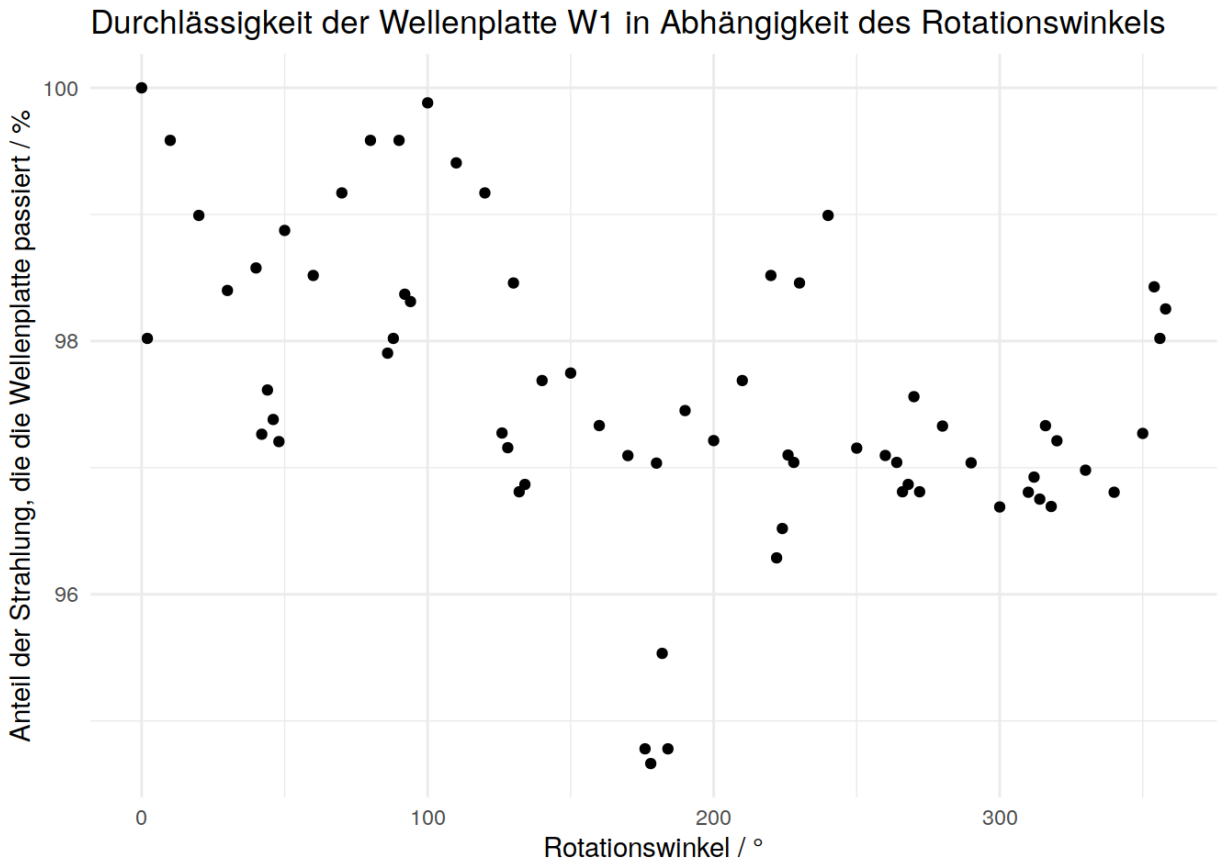
Nullpunkt/Transmissionsverhalten:

- Herunterladen der Messdaten
- Sich wiederholende Daten sind mit NA abgekürzt -> Ersetzen von NA duch die letzte Zahl in der Spalte
- Normalisieren der Messwerte $P_{measured}$ mit der Leistung gemessen ohne Wellenplatte
  $P = \frac{P_{measured}}{P_{background}} \cdot 100\%$

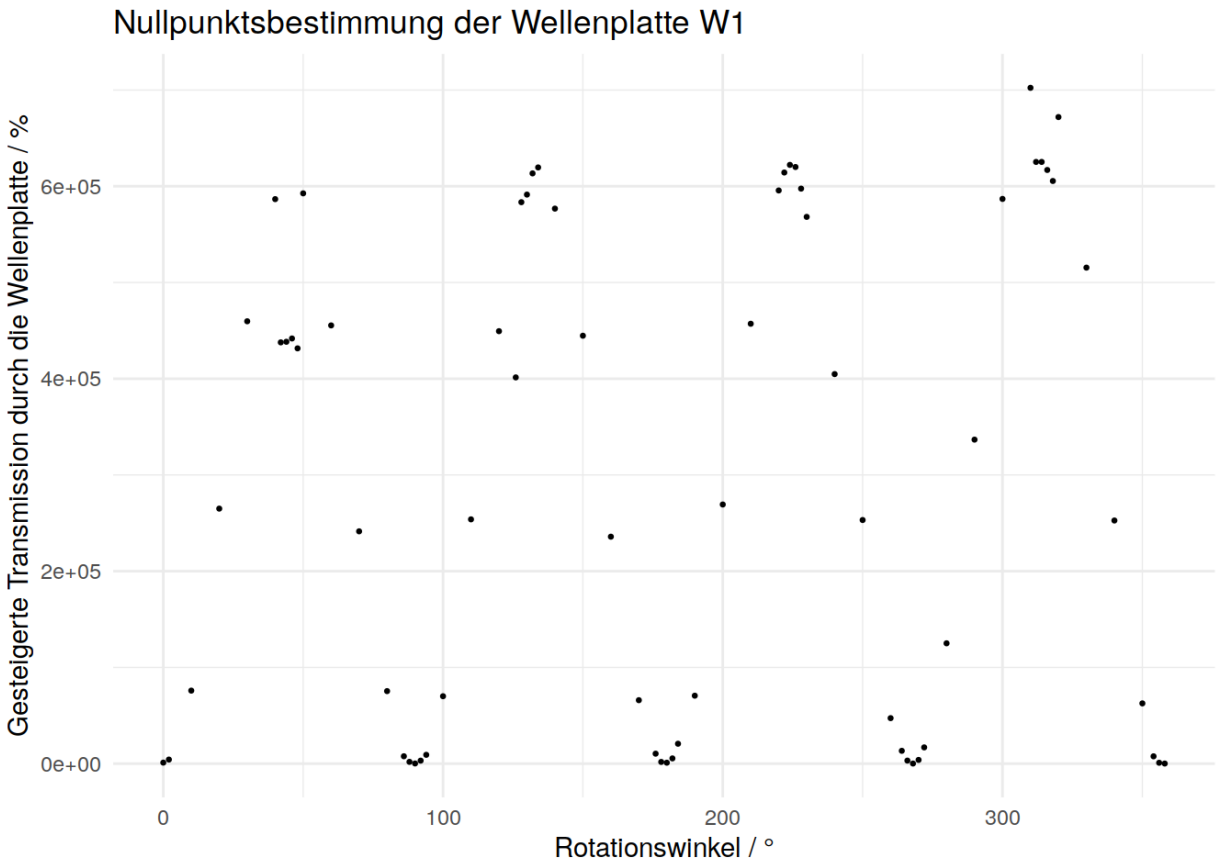
```
#
# Wave Plate W1
#
# Extract data from eLabFTW
W1.data <- GET.elabftw.byselector(27, header = T)[[1]]

# Replace NA values by using the previous value in the vector
fillVector <- function(vector) {
  for (index in seq_along(vector)) {
    if (is.na(vector[index])) { vector[index] <- vector[index-1] }
  }
  return(vector)
}
W1.data$Y2 <- fillVector(W1.data$Y2)
W1.data$Y4 <- fillVector(W1.data$Y4)

# Normalise
W1.data$Y1 <- (W1.data$Y1/W1.data$Y2) *100
W1.data$Y3 <- (W1.data$Y3/W1.data$Y4) *100

#
# Wave Plate W2
#
# Extract data from eLabFTW
W2.data <- GET.elabftw.byselector(28, header = T)[[1]]

# Replace NA values by using the previous value in the vector
fillVector <- function(vector) {
  for (index in seq_along(vector)) {
    if (is.na(vector[index])) { vector[index] <- vector[index-1] }
  }
  return(vector)
}
W2.data$Y2 <- fillVector(W2.data$Y2)
W2.data$Y4 <- fillVector(W2.data$Y4)

# Normalise
W2.data$Y1 <- (W2.data$Y1/W2.data$Y2) *100
W2.data$Y3 <- (W2.data$Y3/W2.data$Y4) *100
```

Wellenplatte W1

```
# Plot Transmissionverhalten
ggplot(data = W1.data, mapping = aes(x = X, y=Y3)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Durchlässigkeit der Wellenplatte W1 in Abhängigkeit des Rotationswinkels",
       x = "Rotationswinkel / °",
       y = "Anteil der Strahlung, die die Wellenplatte passiert / %")
```




```
# Plot Nullpunktsbestimmung
ggplot(data = W1.data, mapping = aes(x=X, y=Y1)) +
  geom_point(size=0.5) +
  theme_minimal() +
  labs(title = "Nullpunktsbestimmung der Wellenplatte W1",
       x = "Rotationswinkel / °",
       y = "Gesteigerte Transmission durch die Wellenplatte / %")
```

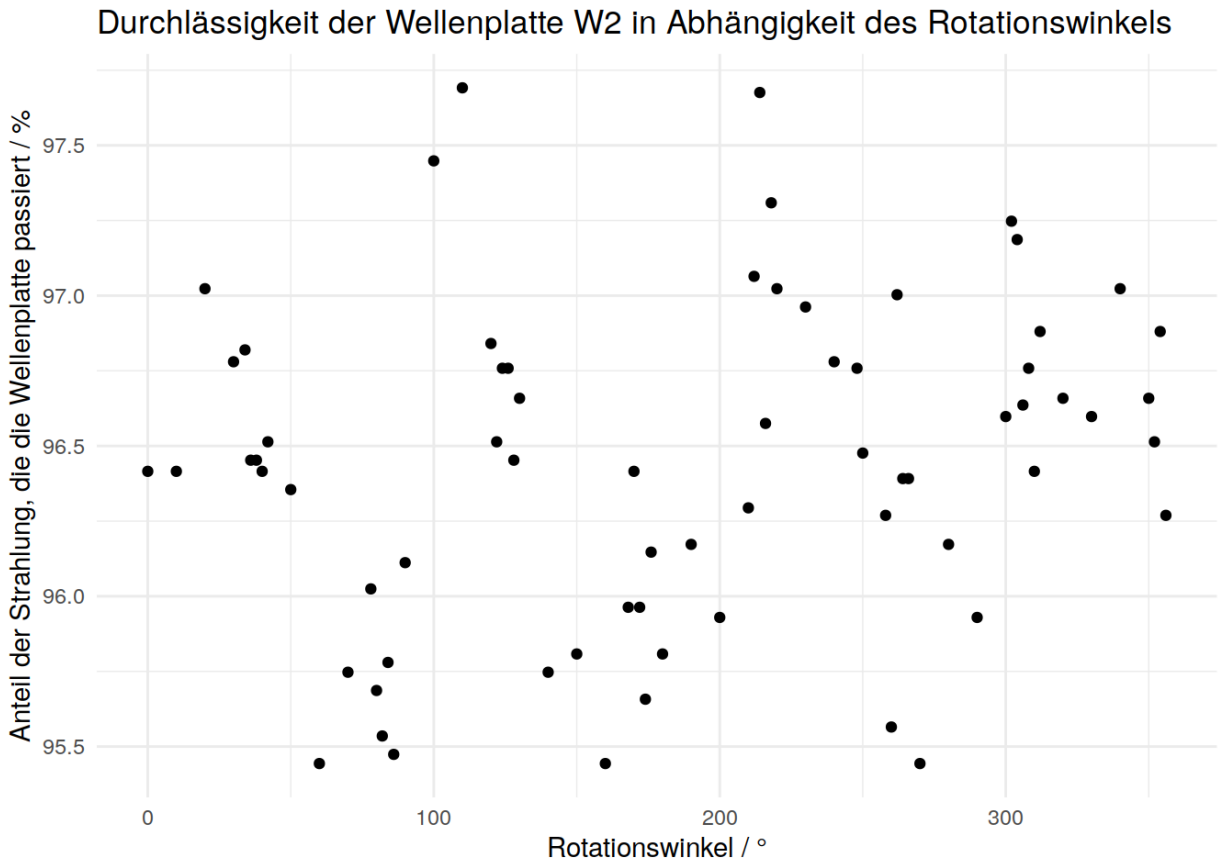


Fazit:

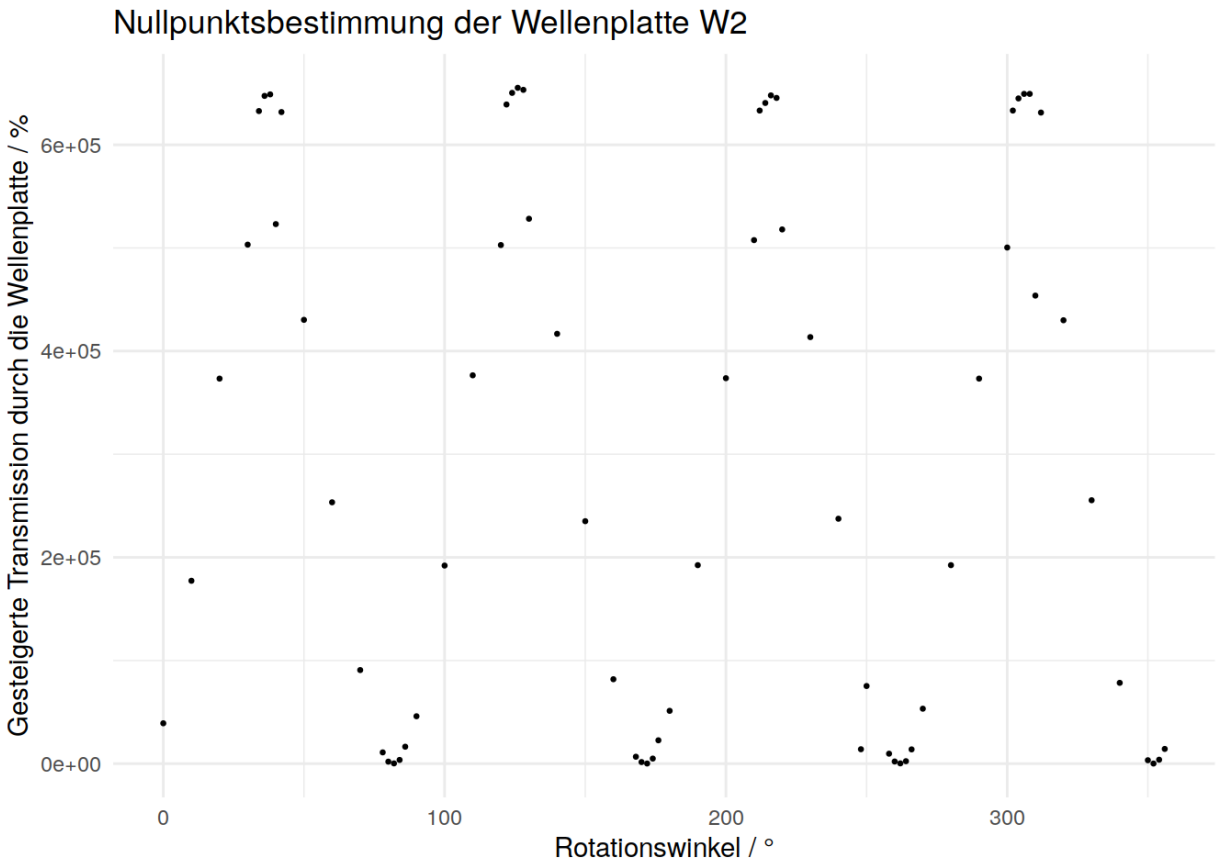
- Wellenplatte rotiert Polarisation um 0° bei: 358°, 90°, 180°, 268°
- Wellenplatte rotiert Polarisation um 90° bei: ~[40°;50°], 134°, 224°, 314°
- W1 steht orientiert zum Fiberport B1
- Wellenplatte transmittiert gut (~98%)
- Transmission schwankt ein wenig (bis zu 2.3%)
- Transmission scheint bei großen Rotationswinkeln zu sinken; es gibt allerdings viele Ausreißer

Wellenplatte W2

```
# Plot Transmissionsverhalten
ggplot(data = W2.data, mapping = aes(x = X, y=Y3)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Durchlässigkeit der Wellenplatte W2 in Abhängigkeit des Rotationswinkels",
       x = "Rotationswinkel / °",
       y = "Anteil der Strahlung, die die Wellenplatte passiert / %")
```




```
# Plot Nullpunktbestimmung
ggplot(data = W2.data, mapping = aes(x=X, y=Y1)) +
  geom_point(size=0.5) +
  theme_minimal() +
  labs(title = "Nullpunktsbestimmung der Wellenplatte W2",
       x = "Rotationswinkel / °",
       y = "Gesteigerte Transmission durch die Wellenplatte / %")
```



- Wellenplatte rotiert Polarisation um 0° bei: 350°, 82°, 172°, 262°
- Wellenplatte rotiert Polarisation um 90° bei: 42°, 128°, 216°, 206°
- Wellenplatte transmittiert sehr gut (~96.5%)
- Transmission schwankt ein wenig (bis zu 2.3%)

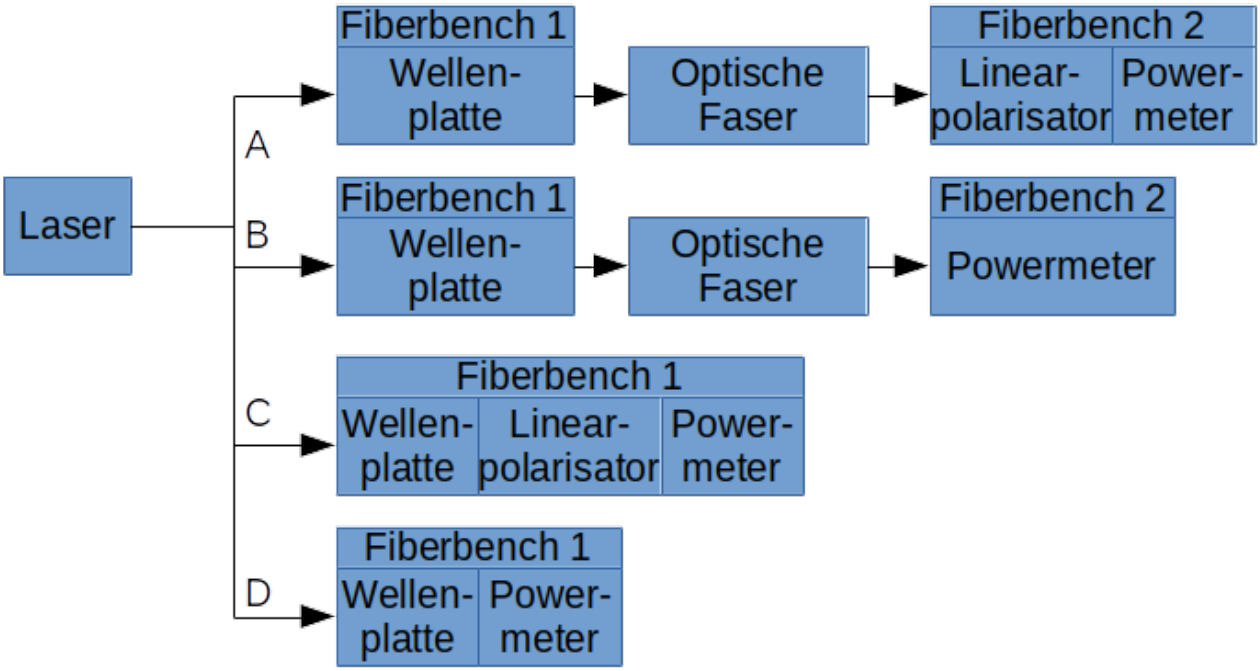
Optische Fasern

Experiment

Es werden drei Experimente durchgeführt:

1. Messung der Stokesvektoren: Die Stokesvektoren werden vor und nach Passieren der optischen Faser für verschiedene Laserpolarisationen bestimmt
2. Messfehler der Stokesvektoren: Die Stokesvektoren werden wiederholt für die selbe initiale Laserpolarisation gemessen, um den Messfehler abzuschätzen
3. Rotation der Polarisationsebene: Es wird gemessen wie stark und in welche Richtung die Faser die Polarisationsebene des Lasers dreht

Messung der Stokesvektoren:



Aufbau Messung Stokesvektoren (A: Stokes hinter der Faser, B: Normierung hinter der Faser, C: Stokes vor der Faser, D: Normierung vor der Faser)

Duchführung:

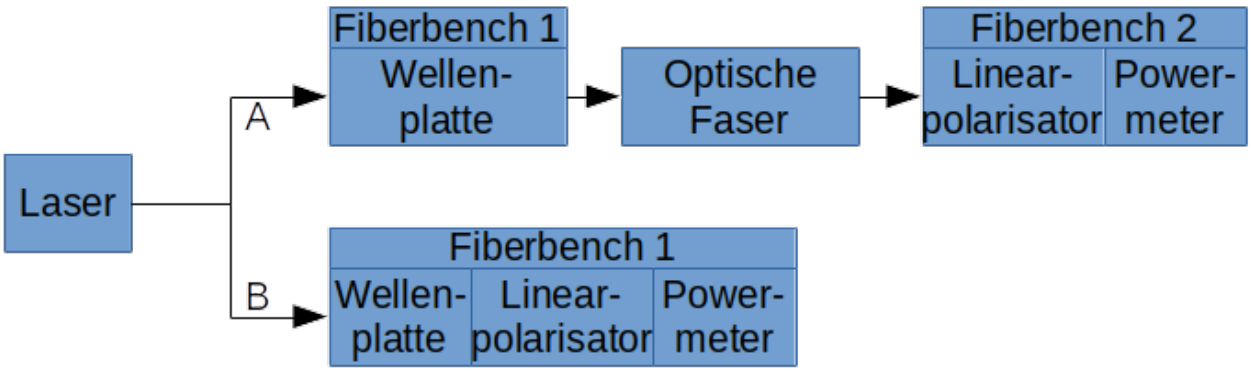
1. Zuerst wird je ein Koordinatensystem für jede Fiberbench definiert (es wird der beschriebene Aufbau verwendet):

- Die Wellenplatte wird für diesen Schritt aus dem Aufbau entfernt
- Für beide Fiberbenchens wird der Linearpolarisator parallel zum Laser ausgerichtet. Die Polarisatorposition wird als 0° definiert.
- Für beide Fiberbenchens wird der Linearpolarisator orhtogonal zum Laser ausgerichtet. Die Polarisatorposition wird als 90° definiert.
- Für beide Fiberbenchens wird der Linearpolarisator erneut parallel zum Laser ausgerichtet. Die Polarisatorposition wird als 180° definiert.

2. Der Stokesvektor wird gemessen, indem für 0°, 45°, 90° und 135° folgende Messung wiederholt wird:

- Die Laserleistung wird analog zur Abbildung für die Aufbauten A, B, C und D gemessen
- Dieser Schritt wird für beliebige Positionen der Wellenplatte wiederholt

Rotation der Polarisationsebene:



Aufbau Rotationsverhalten optischer Fasern (A: hinter der Faser, B: vor der Faser)

Duchführung:

- Für beliebige Positionen der Wellenplatte wird der Linearpolarisator parallel zum Laser gedreht
- Die Polarisatorposition wird vor und nach der optischen Faser notiert

Auswertung

Definition von Funktionen, die die Messdaten auswerten:

- Stokesvektoren vor der Faser $\vec{S}$ und hinter der Faser $\vec{T}$ auf den ersten Stokesparameter S_0 normieren: $\frac{\vec{S}}{S_0}$; $\frac{\vec{T}}{S_0}$
- Berechnen des Polarisationsgrades $\Pi = \frac{\sqrt{S_1^2+S_2^2}}{S_0}$ (letzter Stokesparameter S_3 wird vernachlässigt)
- Berechnen der Änderung des Polarisationsgrades durch die Faser $\Delta\Pi = \frac{\Pi_{post}}{\Pi_{pre}} - 1$
- Berechnen der Änderung der gesamten Laserleistung P_{ges} (Leistung gemessen ohne Linearpolarisator) durch die Faser $\Delta P_{ges} = \frac{P_{ges,post}}{P_{ges,pre}}$):
- Berechnen von Mittelwert, Standardabweichung und Varianz für die wiederholte Messung der selben Laserpolarisation
- Berechnen der Mueller Matrix M für die Fasern. Verwendet werden die Stokesvektoren gemessen vor der Faser $\overrightarrow{S^W}$ und gemessen hinter der Faser $\overrightarrow{T^W}$ (W ist die Position der Wellenplatte, bei der die Stokesvektoren gemessen wurden):

$$M = \begin{pmatrix} m_{00} & m_{01} & m_{02} & 0 \\ m_{10} & m_{11} & m_{12} & 0 \\ m_{20} & m_{21} & m_{22} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- Berechnen der ersten Reihe der Matrix M:

$$\begin{array}{rcccccc} T_0^{0^\circ} & = & m_{00}S_0^{0^\circ} & + & m_{01}S_1^{0^\circ} & + & m_{02}S_2^{0^\circ} \\ T_0^{10^\circ} & = & m_{00}S_0^{10^\circ} & + & m_{01}S_1^{10^\circ} & + & m_{02}S_2^{10^\circ} \\ T_0^{20^\circ} & = & m_{00}S_0^{20^\circ} & + & m_{01}S_1^{20^\circ} & + & m_{02}S_2^{20^\circ} \\ T_0^{30^\circ} & = & m_{00}S_0^{30^\circ} & + & m_{01}S_1^{30^\circ} & + & m_{02}S_2^{30^\circ} \\ \vdots & & & & \vdots & & \end{array}$$

- Berechnen der zweiten Reihe der Matrix M:

$$\begin{array}{rcccccc} T_1^{0^\circ} & = & m_{10}S_0^{0^\circ} & + & m_{11}S_1^{0^\circ} & + & m_{12}S_2^{0^\circ} \\ T_1^{10^\circ} & = & m_{10}S_0^{10^\circ} & + & m_{11}S_1^{10^\circ} & + & m_{12}S_2^{10^\circ} \\ T_1^{20^\circ} & = & m_{10}S_0^{20^\circ} & + & m_{11}S_1^{20^\circ} & + & m_{12}S_2^{20^\circ} \\ T_1^{30^\circ} & = & m_{10}S_0^{30^\circ} & + & m_{11}S_1^{30^\circ} & + & m_{12}S_2^{30^\circ} \\ \vdots & & & & \vdots & & \end{array}$$

- Berechnen der dritten Reihe der Matrix M:

$$\begin{array}{rcccccc} T_2^{0^\circ} & = & m_{20}S_0^{0^\circ} & + & m_{21}S_1^{0^\circ} & + & m_{22}S_2^{0^\circ} \\ T_2^{10^\circ} & = & m_{20}S_0^{10^\circ} & + & m_{21}S_1^{10^\circ} & + & m_{22}S_2^{10^\circ} \\ T_2^{20^\circ} & = & m_{20}S_0^{20^\circ} & + & m_{21}S_1^{20^\circ} & + & m_{22}S_2^{20^\circ} \\ T_2^{30^\circ} & = & m_{20}S_0^{30^\circ} & + & m_{21}S_1^{30^\circ} & + & m_{22}S_2^{30^\circ} \\ \vdots & & & & \vdots & & \end{array}$$

- Erstellen von Funktionen, die Graphen produzieren

```
#
# CALCULATE STOKES VECTORS AND THEIR PROPERTIES
#
# Normalise stokes vector and compute polarisation ratio and such shit
# ASSUMPTION: S3 = 0
process.stokesVec <- function(stokes) {
  # Compute properties of stokes vectors and normalise -> polarisation ratio, ...
  stokes <- lapply(stokes, function(table) {
    # Normalise stokes vectors
    table[,c("S0", "S1", "S2")] <- table[,c("S0", "S1", "S2")] / stokes$PRE$S0

    # Polarisation ratio
    table$polarisation <- sqrt(table$S1^2 + table$S2^2) / table$S0

    # Return result
    return(table)
  })

  # CALCULATE CHANGE OF THE STOKES VECTORS PROPERTIES
  # Change in epsilon, change in polarisation, change in laser intensity
  stokes[["change"]]   <- data.frame("W"                      = stokes$PRE$W,
                                     # How much does the polarisation ratio change?
                                     "change.in.polarisation" = stokes$POST$polarisation /
stokes$PRE$polarisation - 1,
                                     # How much does the intensity of the light change?
                                     "change.in.intensity"    = stokes$POST$I / stokes$PR
E$I
  )

  return(stokes)

}
#
# DO THE STATISTICS
#
# error.stokes contains the data for several identical measurements
# following function will therefore compute statistical properties
# like sd or mean for every column of the tables
do.statistics <- function(error.stokes) {
  lapply(error.stokes, function(table) {
    stats.table <- data.frame( var  = sapply(table, var ),
                               sd   = sapply(table, sd  ),
                               mean = sapply(table, mean)
    )
    return(stats.table)
  }) %>% return


  # TODO: t-Test, Kruskal-Wallis-Test
}



#
# COMPUTE THE MUELLER MATRIX OF THE OPTICAL FIBER
#
# This function takes at least three pairs stokes vectors describing the polarisation of a
laser before (S^PRE) and after (S^POST) interacting
# with an optical fiber (input is the output of process.stokesVec())
muellermatrix <- function(stokes) {
```

```
  # Fit following system of linear equation of three unknown variables (a, b, c) with many
known stokes vector pairs
  # Ignore last stokes parameter
  # (I)   S_0^POST = a_0 * S_0^PRE + b_0 * S_1^PRE + c_0 * S_2^PRE + d_0 * S_3^PRE (S_3^PR
E=0)
  # (II)  S_1^POST = a_1 * S_0^PRE + b_1 * S_1^PRE + c_1 * S_2^PRE + d_1 * S_3^PRE (S_3^PR
E=0)
  # (III) S_2^POST = a_2 * S_0^PRE + b_2 * S_1^PRE + c_2 * S_2^PRE + d_2 * S_3^PRE (S_3^PR
E=0)
  # (IV)  S_3^POST = a_3 * S_0^PRE + b_3 * S_1^PRE + c_3 * S_2^PRE + d_3 * S_3^PRE (S_3^PR
E=0, S_3^POST=0)
  # Build mueller matrix as following:
  # | a_0 b_0 c_0 d_0 |   | a_0 b_0 c_0  0 |
  # | a_1 b_1 c_1 d_1 | = | a_1 b_1 c_1  0 |
  # | a_2 b_2 c_2 d_2 |   | a_2 b_2 c_2  0 |
  # | a_3 b_3 c_3 d_3 |   |  0   0   0   0 |
  matrix( c( limSolve::Solve(as.matrix(stokes$PRE[,c(2,3,4)]), stokes$POST$S0), 0,
             limSolve::Solve(as.matrix(stokes$PRE[,c(2,3,4)]), stokes$POST$S1), 0,
             limSolve::Solve(as.matrix(stokes$PRE[,c(2,3,4)]), stokes$POST$S2), 0,
             0, 0, 0                                                          , 0 ),
          ncol = 4, byrow = T )
}
# Predict the polarisation state after interacting with an optical fiber by using the init
ial
# measured stokes vector and the muller matrix of the fiber
predict.stokesVec <- function(stokes, mueller) {
  # Calculate stokes vectors describing polarisation after the fiber by matrix multiplicat
ion in the mueller formalism
  # Ignore the last stokes parameter
  stokes.post <- apply(stokes$PRE[,c(2,3,4)], 1, function(stokes) {
    mueller[1:3,1:3] %*% ( stokes %>% unlist )
  })
  # Calculate the grade of polarisation
  polarisation <- apply(stokes.post, 2, function(stokes) {
    sqrt(sum(stokes[c(2,3)]^2)) / stokes[1]
  } )

  # Put the result into the inital list of stokes parameters
  stokes$POST.PREDICT <- data.frame(W  = stokes$PRE$W,
                                    S0 = stokes.post[1,],
                                    S1 = stokes.post[2,],
                                    S2 = stokes.post[3,],
                                    I  = NA,
                                    polarisation = polarisation)

  # Calculate the difference between measured and predicted stokes vector
  stokes$PREDICT.ERROR <- data.frame(W = stokes$POST.PREDICT$W,
                                     diff.S0 = stokes$POST$S0 - stokes$POST.PREDICT$S0,
                                     diff.S1 = stokes$POST$S1 - stokes$POST.PREDICT$S1,
                                     diff.S2 = stokes$POST$S2 - stokes$POST.PREDICT$S2,
                                     diff.polarisation = stokes$POST$polarisation - stoke
s$POST.PREDICT$polarisation )

  return(stokes)
}


#
#   PLOT THAT SHIT
```

```
#
# How does the POLARSATION RATIO change RELATIVE to the initial polarisation ratio?
plot.polarisation.change <- function(data,
                                     title = expression(bold("The Depolarising Behaviour O
f <YOUR OPTICAL FIBER>"))
                                     ) {
  # EXPECTED PARAMETERS:
  # data : processedStokesExperiments (output of process.stokesVec)
  # title : expression(bold("The Depolarising Behaviour Of <YOUR OPTICAL FIBER>"))

  # Create the plot
  ggplot(data    = data$change,
         mapping = aes(x = as.factor(W), y = change.in.polarisation*100) ) +
    geom_bar(stat="identity") +
    theme_classic() +
    theme( axis.text = element_text(size=12),
           panel.grid.major.y = element_line("black", size = 0.1),
           panel.grid.minor.y = element_line("grey", size = 0.5) ) +
    labs(title = title,
         x = expression(bold("the have-waveplates angle of rotation "*omega*" / °")),
         y = expression(bold("the relative change in the ratio of polarisation "*Delta*Pi
*" / %")) )

}

# COMPARING POLARISATION RATIOS before and after interacting with the fiber
plot.polarisation <- function(data,
                              statistics,
                              title = expression(bold("The Effect Of An <YOUR OPTICAL FIBE
R> On The Polarisation Ratio "*Pi))
                              ) {
  # PARAMETERS
  # data : processedStokesExperiments (output of process.stokesVec)
  # statistics : processsedErrorMeasurementExperiment (output of process.stokesVec)
  # title : expression(bold("The Effect Of An <YOUR OPTICAL FIBER> On The Polarisation Rat
io "*Pi))

  ggplot(data = data.frame(W = c(data$POST$W, data$PRE$W),
                           polarisation = c(data$POST$polarisation, data$PRE$polarisatio
n),
                           group = c( rep("B_POST", length(data$POST$W)), rep("A_PRE", len
gth(data$PRE$W)) )
                          ),
        mapping=aes(x=as.factor(W), y=polarisation*100, fill=group)) +
    geom_bar(stat="identity", position = "dodge") +
    theme_classic() +
    scale_y_continuous(breaks = seq(from=0, to=110, by=10),
                       expand=c(0,0)) +
    theme(axis.text = element_text(size=12),
          panel.grid.major.y = element_line("black", size = 0.1),
          panel.grid.minor.y = element_line("grey", size = 0.5) ) +
    labs(title = title,
         x     = expression(bold("the have-waveplates angle of rotation "*omega*" / °")),
         y     = expression(bold("the polarisation ratio "*Pi*" / %")),
         fill  = "" ) +
    scale_fill_discrete( labels=c( expression(bold("before")), expression(bold("after")) )
) +
    geom_errorbar(data=data.frame(W     = c(data$PRE$W, data$POST$W) %>% as.factor,
                                  upper = c(data$PRE$polarisation+statistics$POST["polaris
```

```
ation","sd"]*3, data$POST$polarisation+statistics$POST["polarisation","sd"]*3)*100,
                                    lower = c(data$PRE$polarisation-statistics$POST["polaris
ation","sd"]*3, data$POST$polarisation-statistics$POST["polarisation","sd"]*3)*100,
                                    group = c( rep("A_PRE", length(data$POST$W)), rep("B_POS
T", length(data$PRE$W)) ),
                                    polarisation = c(data$PRE$polarisation, data$POST$polari
sation) ),
                  mapping = aes(x=W, ymin = lower, ymax=upper, group=group),
                  position = "dodge"
    )
}

# CHANGE in LASER POWER due to optical fiber
plot.intensity.change <- function(data,
                                    title = expression(bold("The Transmittance Of <YOUR OPTI
CAL FIBER>"))
                                    ) {
  # EXPECTED PARAMETERS:
  # data : processedStokesExperiments (output of process.stokesVec)
  # title : expression(bold("The Depolarising Behaviour Of <YOUR OPTICAL FIBER>"))

  ggplot( data    = data$change,
          mapping = aes(x = as.factor(W), y = change.in.intensity*100) ) +
    geom_bar(stat="identity") +
    theme_classic() +
    theme(axis.text = element_text(size=12),
          panel.grid.major.y = element_line("black", size = 0.1),
          panel.grid.minor.y = element_line("grey", size = 0.5) ) +
    labs(title = title,
         x = expression(bold("the have-waveplates angle of rotation "*omega*" / °")),
         y = expression(bold("the transmitted part of the laser P"[trans]*" / %")) )
}


# COMPARING LASER POWER before and after interacting with the fiber
plot.intensity <- function(data,
                           title = expression(bold("The Effect Of <YOUR OPTICAL FIBER> On
The Lasers Power "*P))
                          ) {
  # EXPECTED PARAMETERS:
  # data : processedStokesExperiments (output of process.stokesVec)
  # title : expression(bold("The Depolarising Behaviour Of <YOUR OPTICAL FIBER>"))

  ggplot(data = data.frame(W = c(data$POST$W, data$PRE$W),
                           intensity = c(data$POST$I, data$PRE$I) /  data$PRE$I,
                           group = c( rep("B_POST", length(data$POST$W)), rep("A_PRE", len
gth(data$PRE$W)) )
                          ),
         mapping=aes(x=as.factor(W), y=intensity*100, fill=group)) +
    geom_bar(stat="identity", position = "dodge") +
    theme_classic() +
    theme(axis.text = element_text(size=12),
          panel.grid.major.y = element_line("black", size = 0.1),
          panel.grid.minor.y = element_line("grey", size = 0.5) ) +
    labs(title = title,
         x     = expression(bold("have-waveplates angle of rotation "*omega*" / °")),
         y     = expression(bold("normalised laser power "*P*" / %")),
         fill  = "" ) +
    scale_fill_discrete( labels=c( expression(bold("before")), expression(bold("after")) )
```

```
)
}


# How does the fiber influence the PLANE OF POLARISATIONS ORIENTATION?
plot.plane.rotation <- function(data,
                                title = expression(bold("Impact Of The <YOUR FIBER> On The
Orientation Of The Plane Of Polarisation"))
) {
  # EXPECTED PARAMETERS:
  # data : elabFTW table of angle dependent rotation behavior of optical fibers
  # title : expression(bold("The Impact Of The <YOUR FIBER> On The Orientation Of The Plan
e Of Polarisation"))

  ggplot( data = data[!is.na(data$X),] ) +
    geom_abline( mapping = aes(intercept = 0, slope = 2, color="ideal waveplate") ) +
    geom_abline( mapping = aes(intercept = 0, slope = -2, color="ideal waveplate") ) +
    geom_point( mapping = aes(x = X, y = Y1-Y1[X==0], color = "after") ) +
    geom_point( mapping = aes(x = X, y = Y2-Y2[X==0], color = "before") ) +
    theme_classic() +
    theme(panel.grid.major = element_line("black", size=0.1),
          panel.grid.minor = element_line("grey", size=0.1) ) +
    scale_x_continuous(breaks = seq(from=-20, to=300, by=20) ) +
    scale_y_continuous(breaks = seq(from=-360, to=360, by=90) ) +
    labs(title = title,
         x = expression(bold("rotation waveplate / °")),
         y = expression(bold("rotation linear polariser / °")),
         color = "" )
}
```

Nutzung der eben definierten Funktionen:

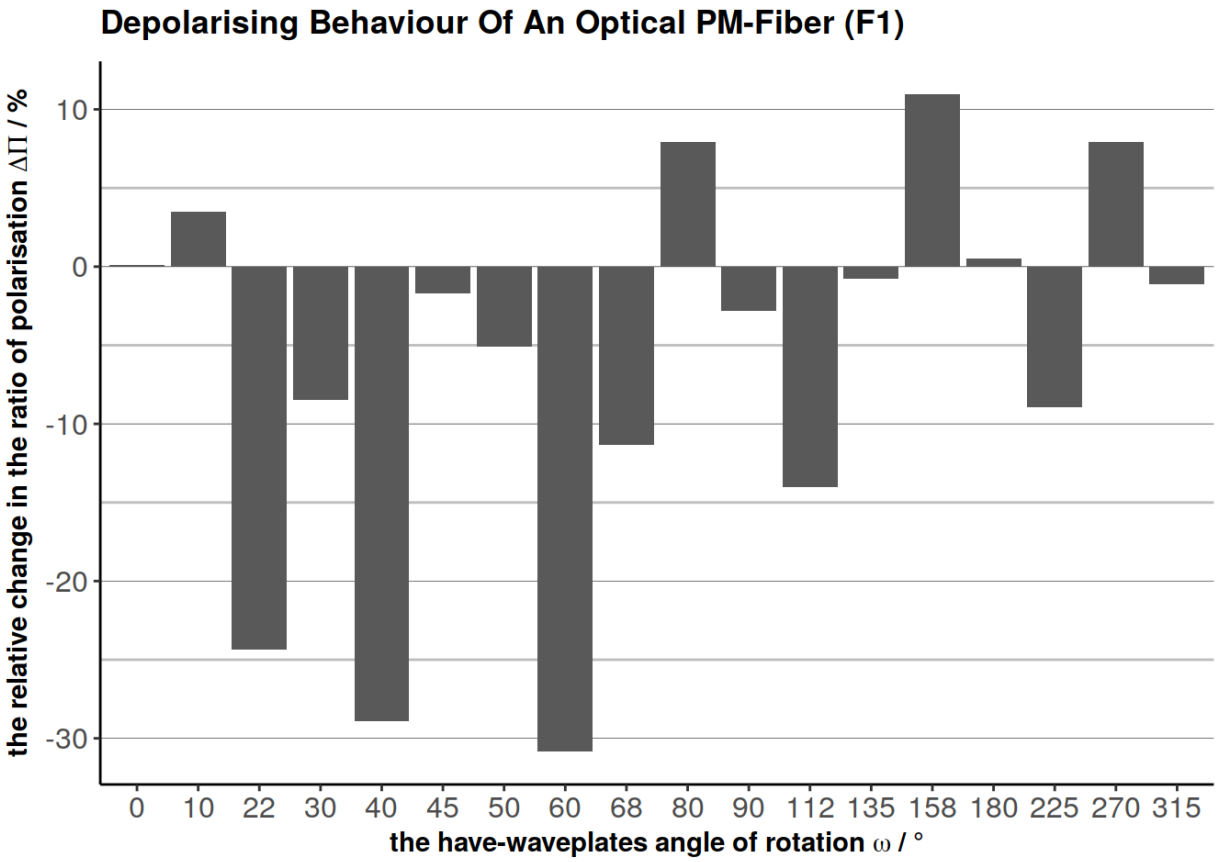
- Herunterladen der Messdaten
- .csv-Dateien auslesen und den Mittelwert (unter Vernachlässigung der ersten und letzten 10%-Quantille) der gemessenen Leistungen nehmen (Funktionen parseTable.elabftw and qmean erledigen das)
- Berechnen der Stokesvektoren $\vec{S}$ aus dem gemessenen Laserleistungen $P_{0^\circ}; P_{45^\circ}; P_{90^\circ}; P_{135^\circ}$ (Definiert in RHotStuff): $\vec{S} = (P_{0^\circ} + P_{90^\circ}; P_{0^\circ} - P_{90^\circ}; P_{45^\circ} - P_{135^\circ}; 0)$ (letzter Stokesparameter wird vernachlässigt)
- Polarisationsgrad berechnen (vgl. process.stokesVec() im vorherigen code chunk)
- Messfehler berechnen (Mittelwert, Standardabweichung, Varianz für wiederholte Messung -> do.statistics() definiert in vorherigen code chunk)

```
# F1 : POLARISATION-MAINTAING-FIBER

# FETCH data from elabftw
# stokes vector
F1.data.elab <- lapply(c(58, 67, 68), function(experimentID) {
  GET.elabftw.bycaption(experimentID, header=T, outputHTTP=T) %>%
    parseTable.elabftw(., func=function(x) qmean(x[,4], 0.8, na.rm=T, inf.rm=T),
                       header=T, skip=14, sep=";")
}) %>% better.rbind(., sort.byrow=1)

# Get the measurements for the error estimation
F1.error.elab <- GET.elabftw.bycaption(66, header=T, outputHTTP=T) %>%
                  parseTable.elabftw(., func=function(x) qmean(x[,4], 0.8, na.rm=T, inf.rm
=T),
                                     header=T, skip=14, sep=";")

# COMPUTE stokes vectors and do statistics on the error estimations
F1.data.stokes <- getStokes.from.expData(F1.data.elab)  %>% process.stokesVec
F1.error.stats <- getStokes.from.expData(F1.error.elab) %>% process.stokesVec %>% do.stati
stics

# COMPUTE the mueller matrix of the fiber, using the stokes vectors measured before and af
ter the fiber
F1.muellermatrix <- muellermatrix(F1.data.stokes)
# PREDICT the stokes vectors after the fiber from the mueller matrix and the stokes vector
s measured before the fiber
F1.data.stokes   <- predict.stokesVec(F1.data.stokes, F1.muellermatrix)


# F2 : SINGLE-MODE-FIBER

# FETCH data from elabftw
# stokes vectors
F2.data.elab <- lapply(c(69, 70), function(experimentID) {
  GET.elabftw.bycaption(experimentID, header=T, outputHTTP=T) %>%
    parseTable.elabftw(., func=function(x) qmean(x[,4], 0.8, na.rm=T, inf.rm=T),
                       header=T, skip=14, sep=";")
}) %>% better.rbind(., sort.byrow=1)
# Rotation of plane of polarisation
F2.rotation.elab <- GET.elabftw.bycaption(72, header=T)[[1]]

# Get the measurements for the error estimation
F2.error.elab <- GET.elabftw.bycaption(71, header=T, outputHTTP=T) %>%
                  parseTable.elabftw(., func=function(x) qmean(x[,4], 0.8, na.rm=T, inf.rm
=T),
                                     header=T, skip=14, sep=";")


# COMPUTE stokes vectors and do statistics on the error estimations
F2.data.stokes <- getStokes.from.expData(F2.data.elab)  %>% process.stokesVec
F2.error.stats <- getStokes.from.expData(F2.error.elab) %>% process.stokesVec %>% do.stati
stics

# COMPUTE the mueller matrix of the fiber, using the stokes vectors measured before and af
ter the fiber
F2.muellermatrix <- muellermatrix(F2.data.stokes)
# PREDICT the stokes vectors after the fiber from the mueller matrix and the stokes vector
s measured before the fiber
F2.data.stokes   <- predict.stokesVec(F2.data.stokes, F2.muellermatrix)
```

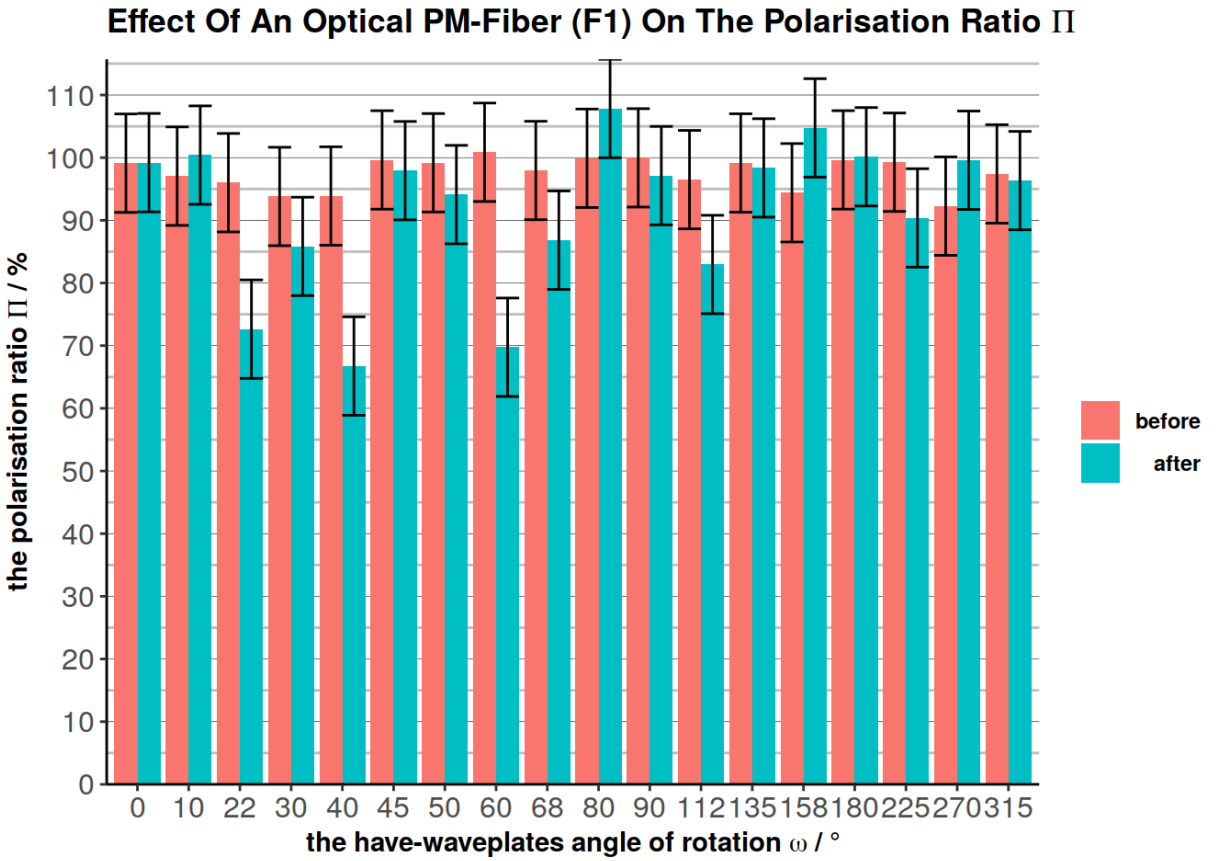
```
# FIBER F3 : MULTI-MODE-FIBER

# FETCH data from elabftw
# stokes vectors
F3.data.elab <- GET.elabftw.bycaption(74, header=T, outputHTTP=T) %>%
                  parseTable.elabftw(., func=function(x) qmean(x[,4], 0.8, na.rm=T, inf.rm
=T),
                                     header=T, skip=14, sep=";")
# Rotation of plane of polarisation
F3.rotation.elab <- GET.elabftw.bycaption(73, header=T)[[1]]

# Get the measurements for the error estimation
F3.error.elab <- GET.elabftw.bycaption(75, header=T, outputHTTP=T) %>%
                  parseTable.elabftw(., func=function(x) qmean(x[,4], 0.8, na.rm=T, inf.rm
=T),
                                     header=T, skip=14, sep=";")


# COMPUTE stokes vectors and do statistics on the error estimations
F3.data.stokes <- getStokes.from.expData(F3.data.elab)  %>% process.stokesVec
F3.error.stats <- getStokes.from.expData(F3.error.elab) %>% process.stokesVec %>% do.stati
stics

# COMPUTE the mueller matrix of the fiber, using the stokes vectors measured before and af
ter the fiber
F3.muellermatrix <- muellermatrix(F3.data.stokes)
# PREDICT the stokes vectors after the fiber from the mueller matrix and the stokes vector
s measured before the fiber
F3.data.stokes   <- predict.stokesVec(F3.data.stokes, F3.muellermatrix)
```

PM-Faser F1

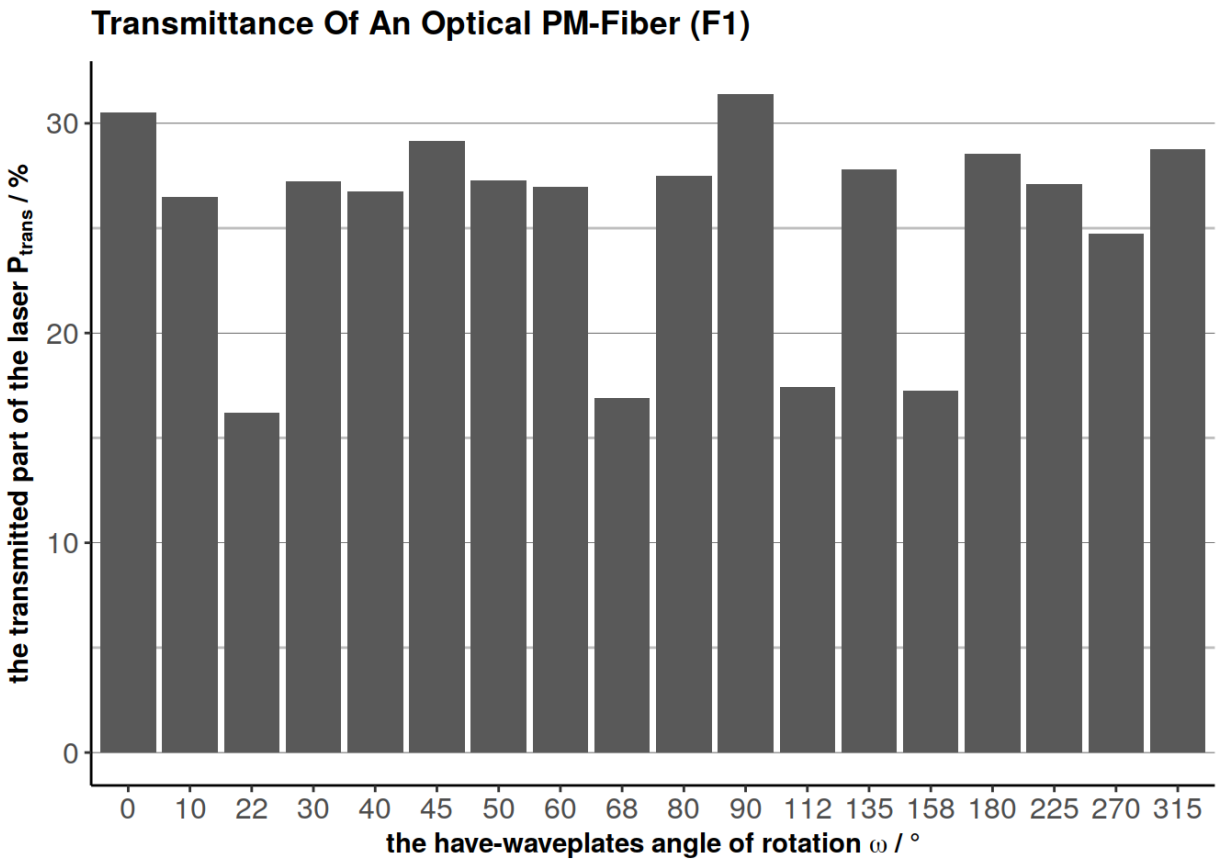
```
# How does the polarisation ratio change relative to the initial polarisation ratio?
plot.polarisation.change(data  = F1.data.stokes,
                         title = expression(bold("Depolarising Behaviour Of An Optical PM-
Fiber (F1)") )
)
```




```
# COMPARING POLARISATION RATIOS before and after interacting with the fiber
plot.polarisation(data       = F1.data.stokes,
                  statistics = F1.error.stats,
                  title      = expression(bold("Effect Of An Optical PM-Fiber (F1) On The
Polarisation Ratio "*Pi))
)
```



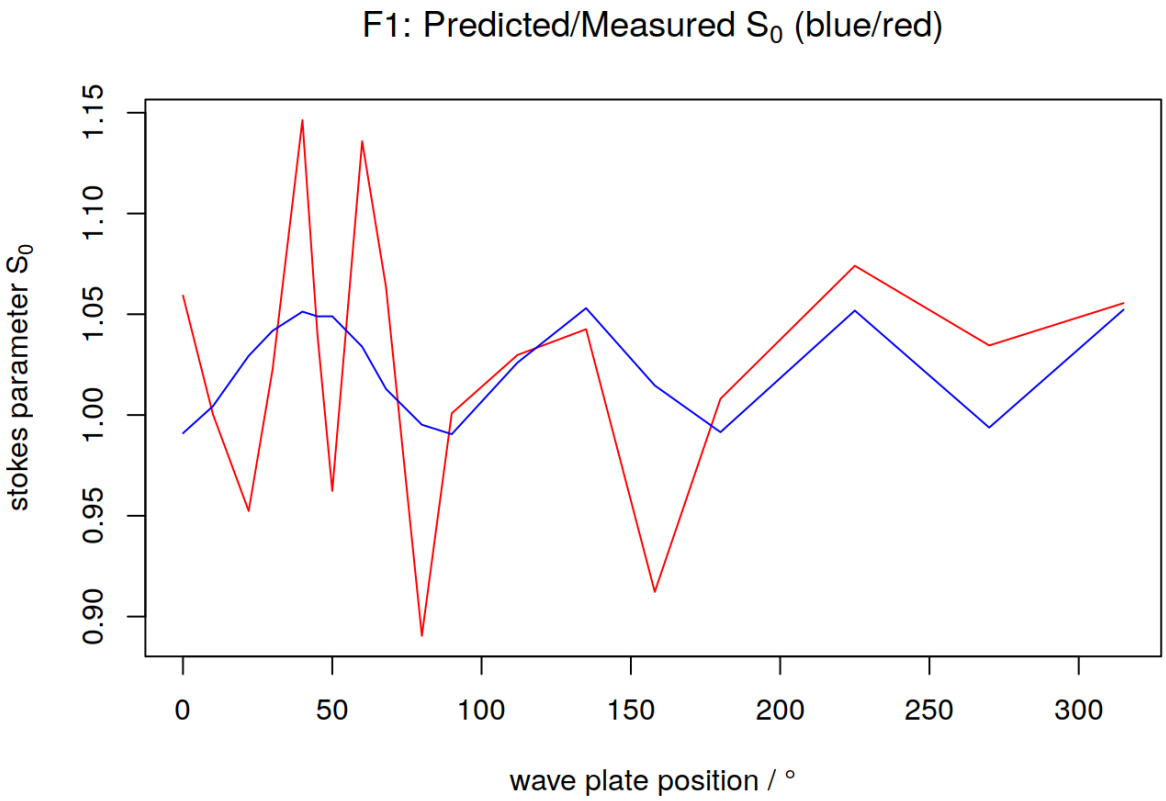
```
# CHANGE in LASER POWER due to optical fiber
plot.intensity.change(data  = F1.data.stokes,
                      title = expression(bold("Transmittance Of An Optical PM-Fiber (F
1)"))
)
```



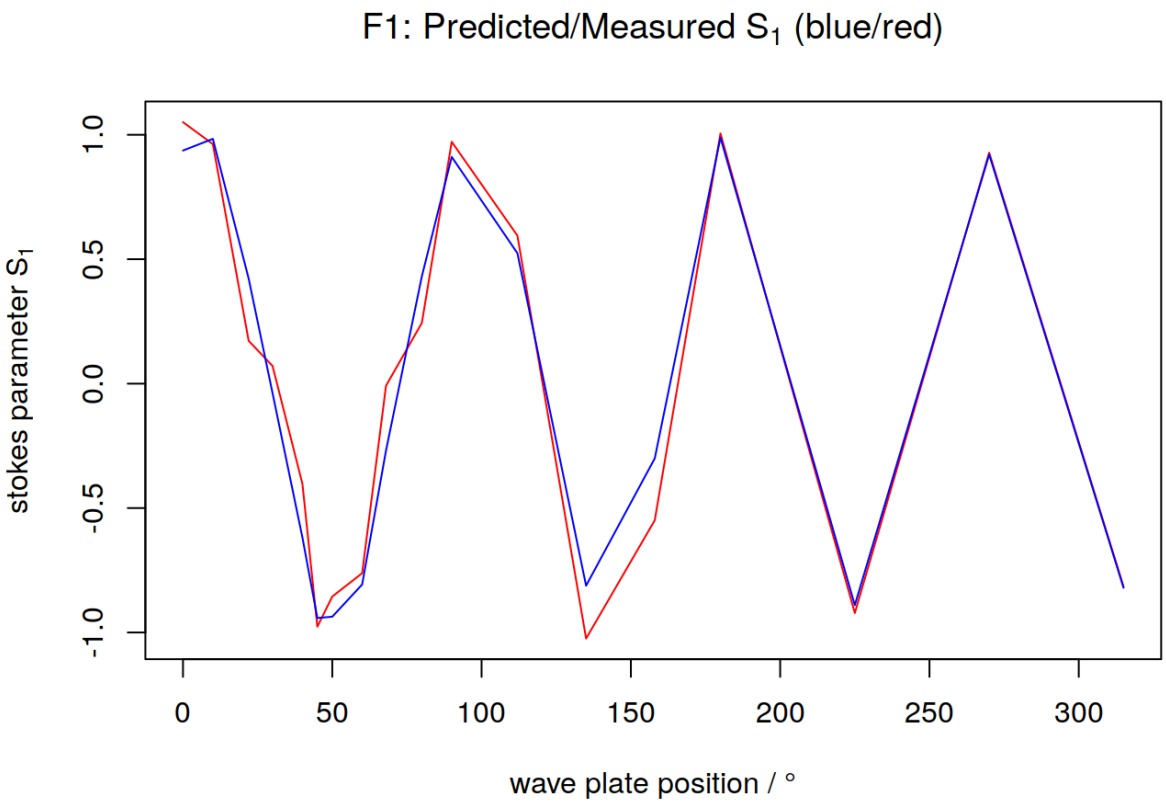

```
# Print calculated mueller matrix
print(F1.muellermatrix)
```

```
##            [,1]        [,2]        [,3] [,4]
## [1,] 1.02194983 -0.03089329 0.006487982    0
## [2,] 0.05730866  0.91736154 0.416441989    0
## [3,] 0.08553676 -0.28682824 0.250438862    0
## [4,] 0.00000000  0.00000000 0.000000000    0
```

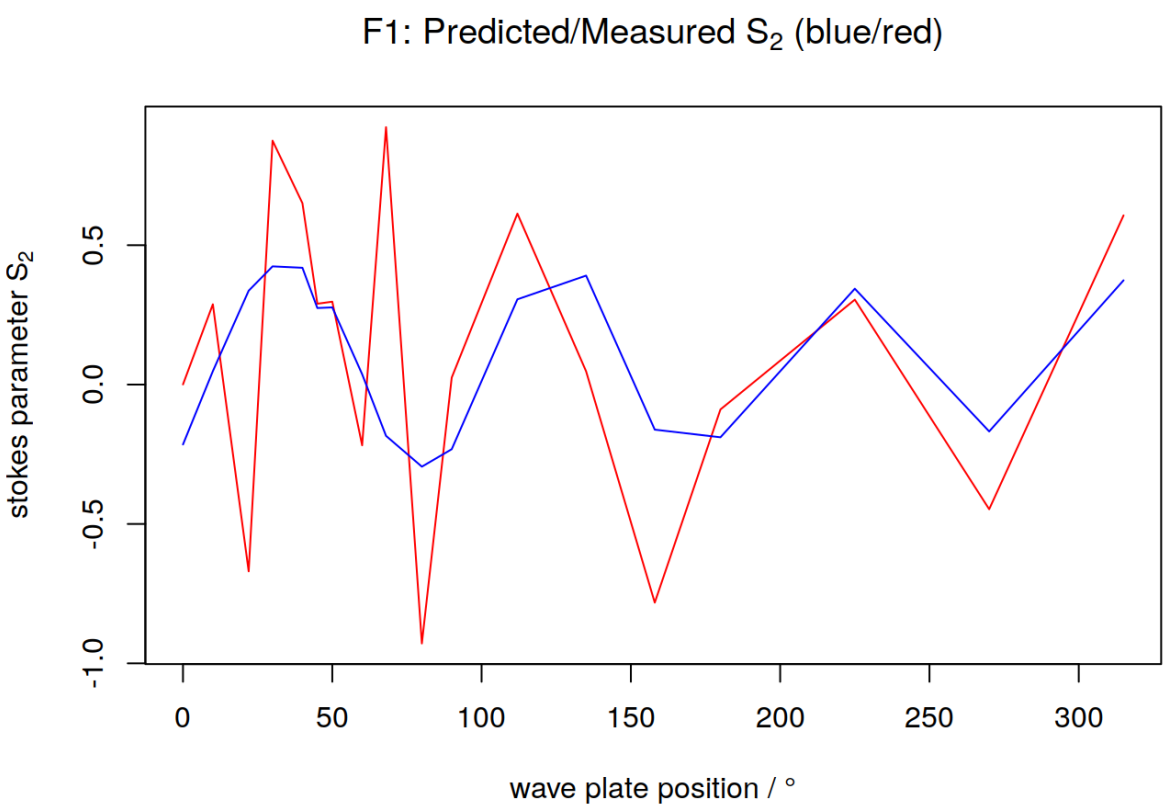
```
# Plot and compare the PREDICTED and MEASURED STOKES parameters
# S0
plot(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST$S0, col="red", type="l",
     main = expression("F1: Predicted/Measured S"[0]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[0]))
lines(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST.PREDICT$S0, col="blue")
```



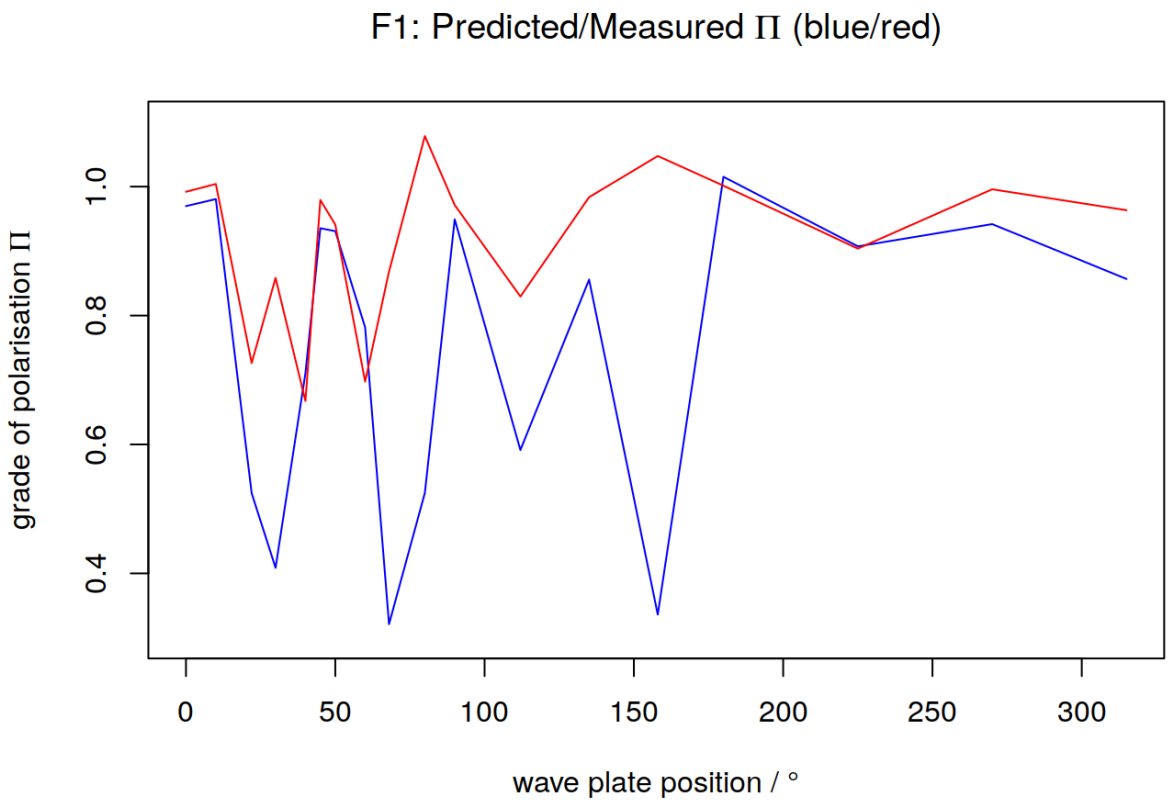

```
# S1
plot(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST$S1, col="red", type="l",
     main = expression("F1: Predicted/Measured S"[1]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[1]))
lines(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST.PREDICT$S1, col="blue")
```



```
# S2
plot(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST$S2, col="red", type="l",
     main = expression("F1: Predicted/Measured S"[2]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[2]))
lines(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST.PREDICT$S2, col="blue")
```




```
# Polarisation ratio
plot(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST.PREDICT$polarisation, col="blue",
type="l",
     main = expression("F1: Predicted/Measured "*Pi*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("grade of polarisation "*Pi),
     ylim = c(0.3, 1.1))
lines(x = F1.data.stokes$POST$W, y = F1.data.stokes$POST$polarisation, col="red")
```

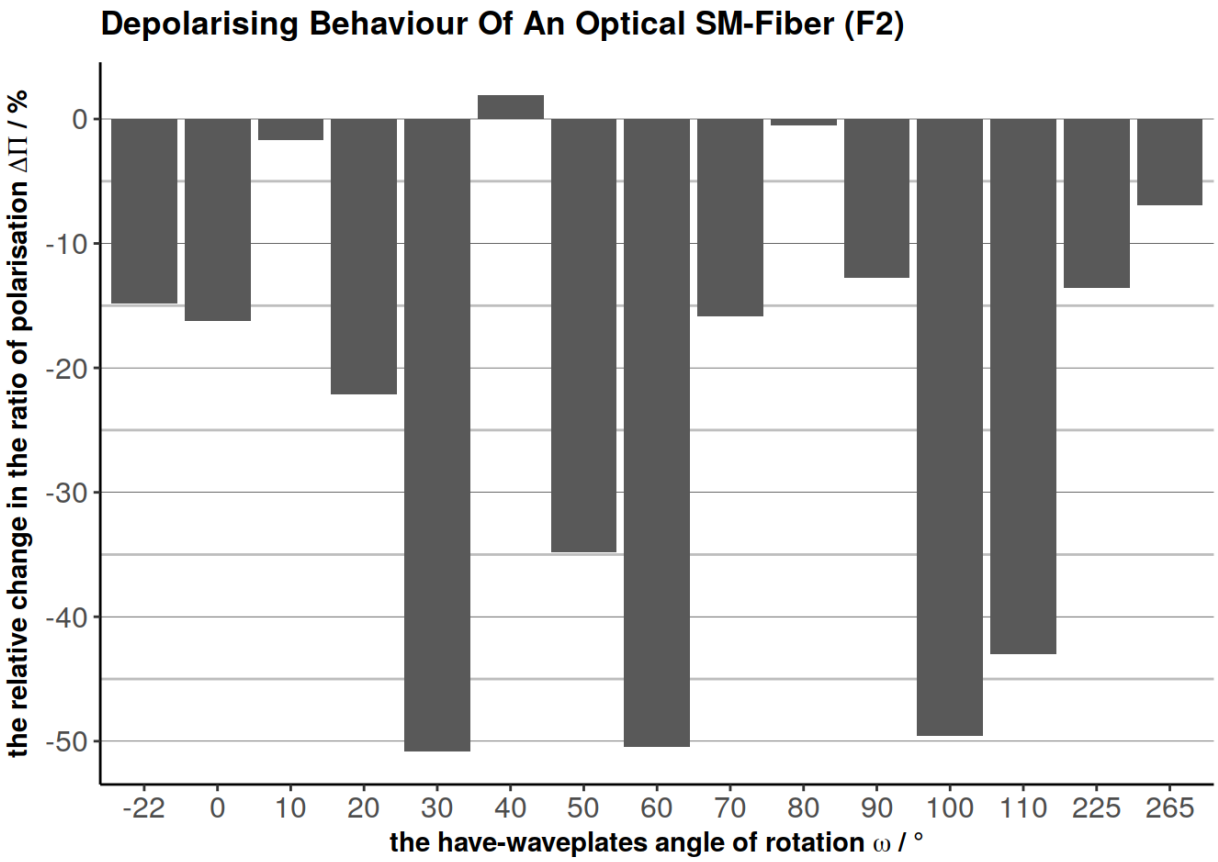


Fazit:

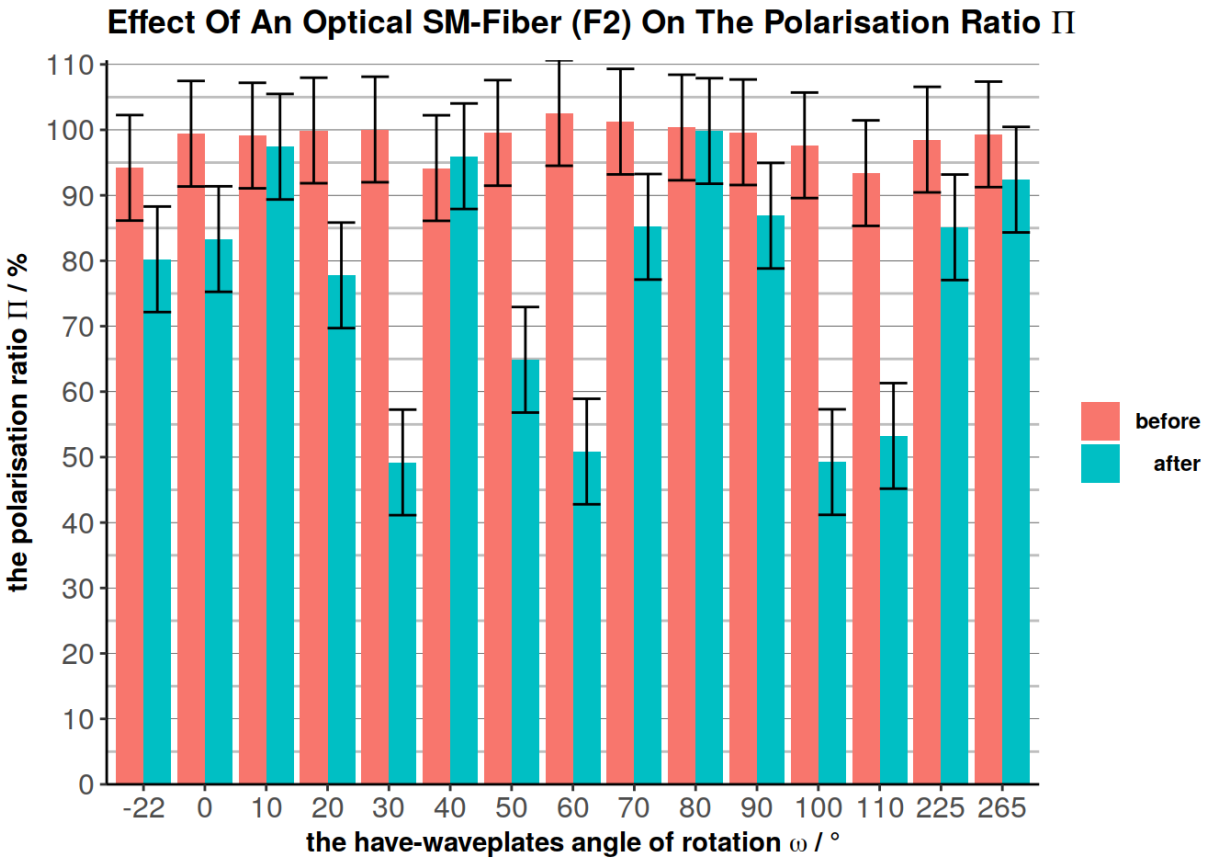
- Polarisation wird erhalten, wenn Wellenplatte bei 0°, 45°, 90° oder einer entsprechenden größeren Winkel (z.B. 180°) steht
- In der Nähe der polarisationserhaltenen Winkel wird der Polarisationsgrad vergrößert
- In allen anderen Fällen wird der Polarisationsgrad verringert
- Anhand der Änderungen im Polarisationsgrad wird deutlich: Die PM-Faser leitet nur zwei Moden. Die erste bei 0°, die zweite bei 90°
- Die PM-Faser kann nicht für beliebige Polarisationszustände verwendet werden
- Die Faser lässt 15% bis 30% des Lasers durch
- Wie erwartet: Bei 0° und 90° resultierender Polarisationsgrad sehr gut ($\widehat{=}\ n \cdot 45°$ Wellenplattenrotation). Dazwischen depolarisiert die Faser, weil in andere Moden gepumpt wird
- Bei 0° und 90° Transmission sehr gut ($\widehat{=}\ n \cdot 45°$ Wellenplattenrotation). Dazwischen gibt es Kopplungsverluste
- Müllermatrix sagt nur den zweiten Stokesparameter korrekt voraus; Polarisationsgrad und andere zwei Stokesparameter sind falsch
- Warum ist die Müllermatrix falsch? Für F3 hat es funktioniert. Muss das Experiment wiederholt werden? Muss der geschätzte Messfehler berücksichtigt werden?

SM-Faser F2

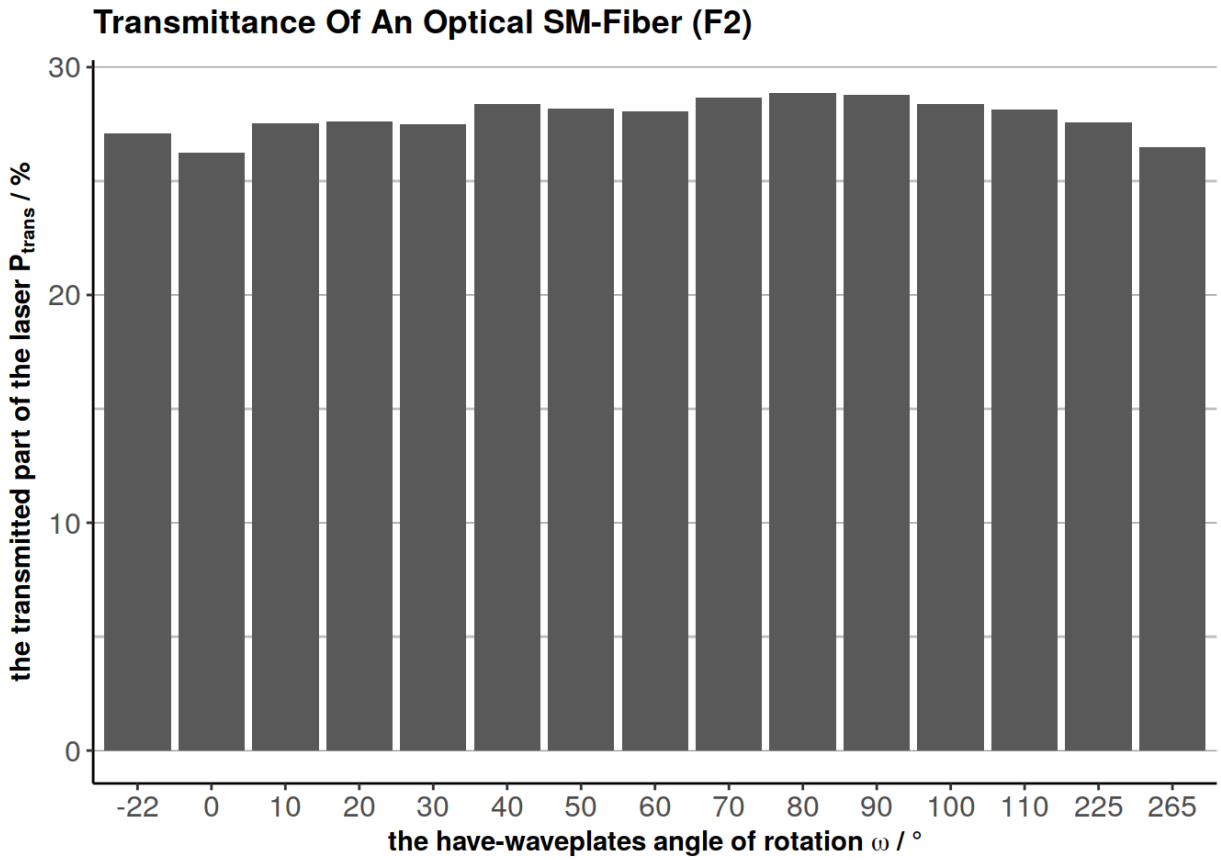
```
# How does the polarisation ratio change relative to the initial polarisation ratio?
plot.polarisation.change(data  = F2.data.stokes,
                         title = expression(bold("Depolarising Behaviour Of An Optical SM-
Fiber (F2)") )
)
```




```
# COMPARING POLARISATION RATIOS before and after interacting with the fiber
plot.polarisation(data       = F2.data.stokes,
                  statistics = F2.error.stats,
                  title      = expression(bold("Effect Of An Optical SM-Fiber (F2) On The
Polarisation Ratio "*Pi))
)
```

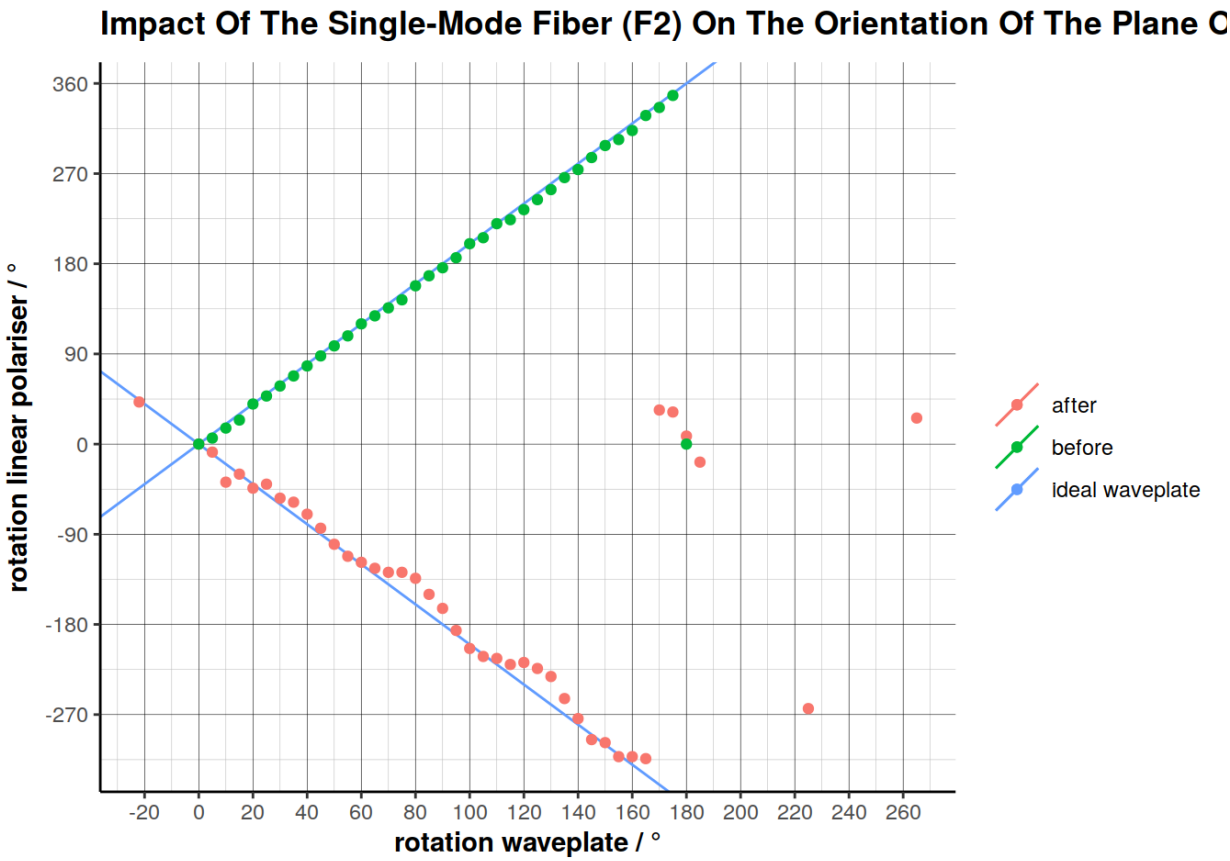


```
# CHANGE in LASER POWER due to optical fiber
plot.intensity.change(data  = F2.data.stokes,
                      title = expression(bold("Transmittance Of An Optical SM-Fiber (F
2)"))
)
```




```
# How does the fiber influence the PLANE OF POLARISATIONS ORIENTATION
plot.plane.rotation(F2.rotation.elab,
                    title = expression(bold("Impact Of The Single-Mode Fiber (F2) On The O
rientation Of The Plane Of Polarisation"))
)
```

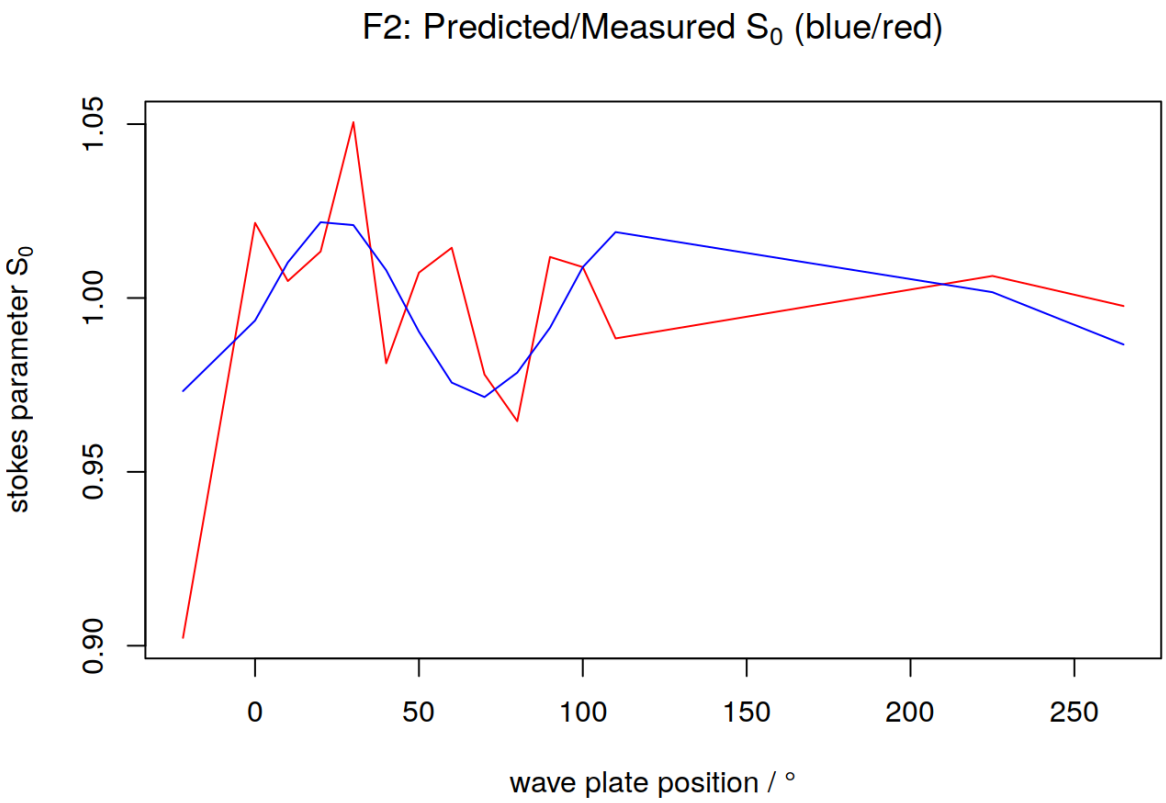
```
## Warning: Removed 4 rows containing missing values (geom_point).
```



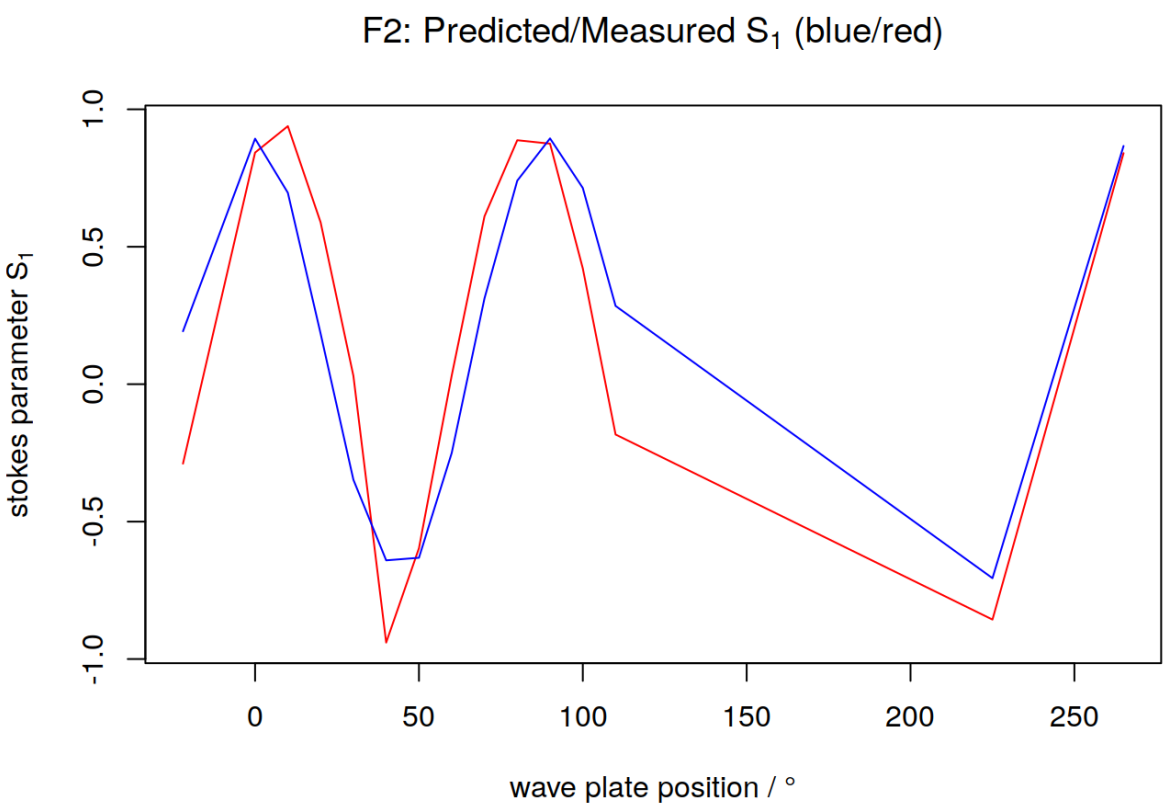

```
# Print calculated mueller matrix
print(F2.muellermatrix)
```

```
##            [,1]         [,2]        [,3] [,4]
## [1,] 0.99730529 -0.004265521  0.02523766    0
## [2,] 0.09017625  0.808027947 -0.00876818    0
## [3,] 0.05096364  0.046154634 -0.57918936    0
## [4,] 0.00000000  0.000000000  0.00000000    0
```

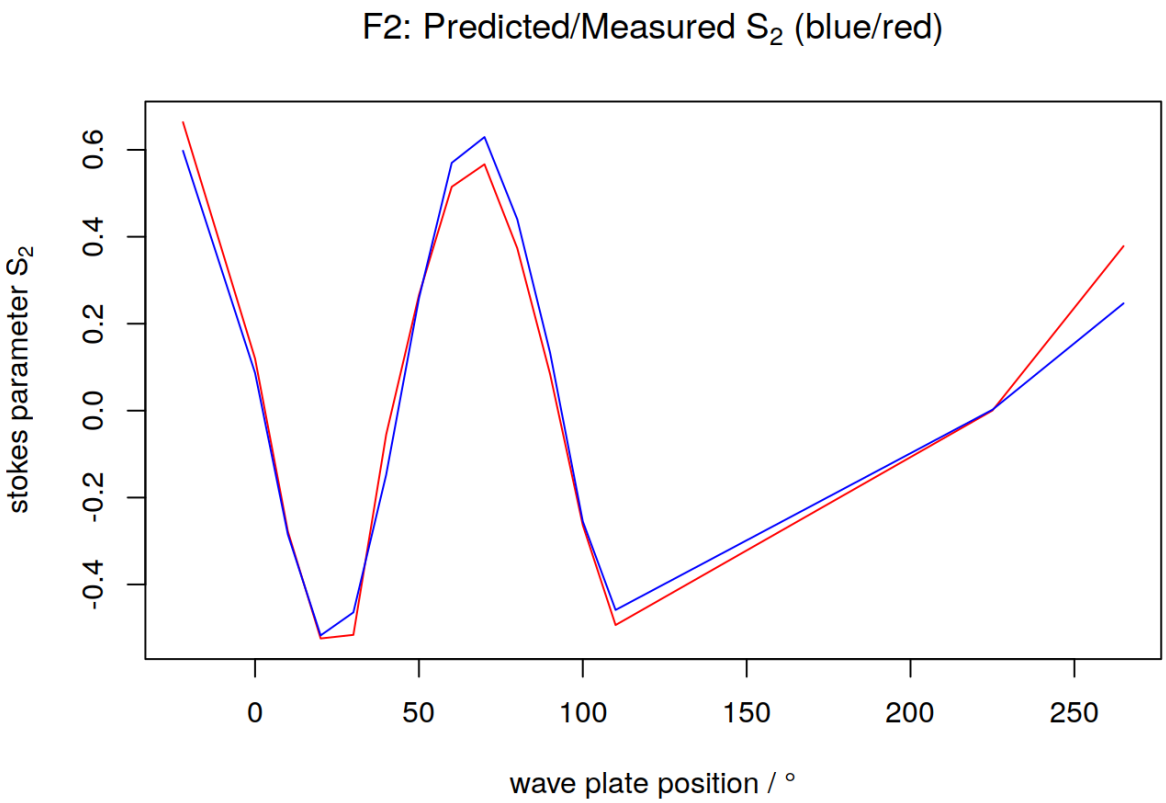
```
# Plot and compare the PREDICTED and MEASURED STOKES parameters
# S0
plot(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST$S0, col="red", type="l",
     main = expression("F2: Predicted/Measured S"[0]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[0]))
lines(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST.PREDICT$S0, col="blue")
```



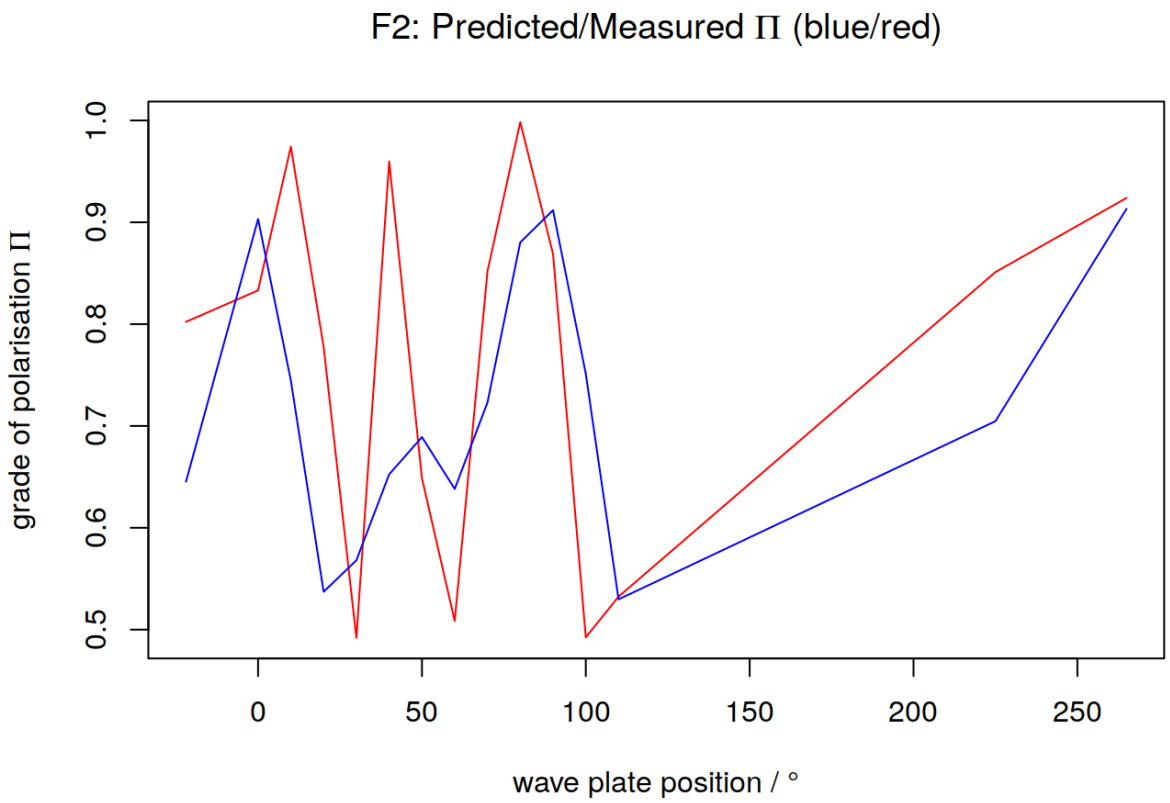

```
# S1
plot(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST$S1, col="red", type="l",
     main = expression("F2: Predicted/Measured S"[1]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[1]))
lines(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST.PREDICT$S1, col="blue")
```



```
# S2
plot(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST$S2, col="red", type="l",
     main = expression("F2: Predicted/Measured S"[2]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[2]))
lines(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST.PREDICT$S2, col="blue")
```




```
# Polarisation ratio
plot(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST$polarisation, col="red", type="l",
     main = expression("F2: Predicted/Measured "*Pi*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("grade of polarisation "*Pi))
lines(x = F2.data.stokes$POST$W, y = F2.data.stokes$POST.PREDICT$polarisation, col="blue")
```

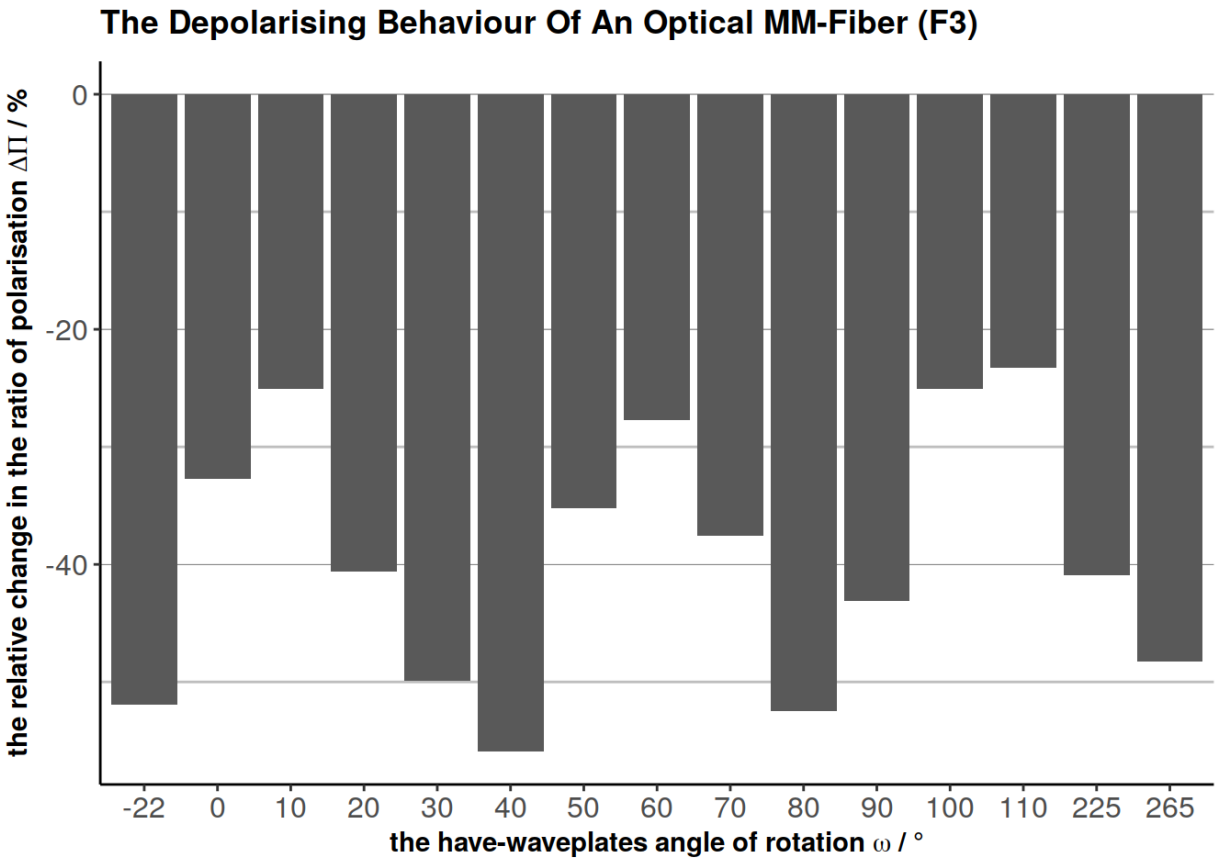


Fazit:

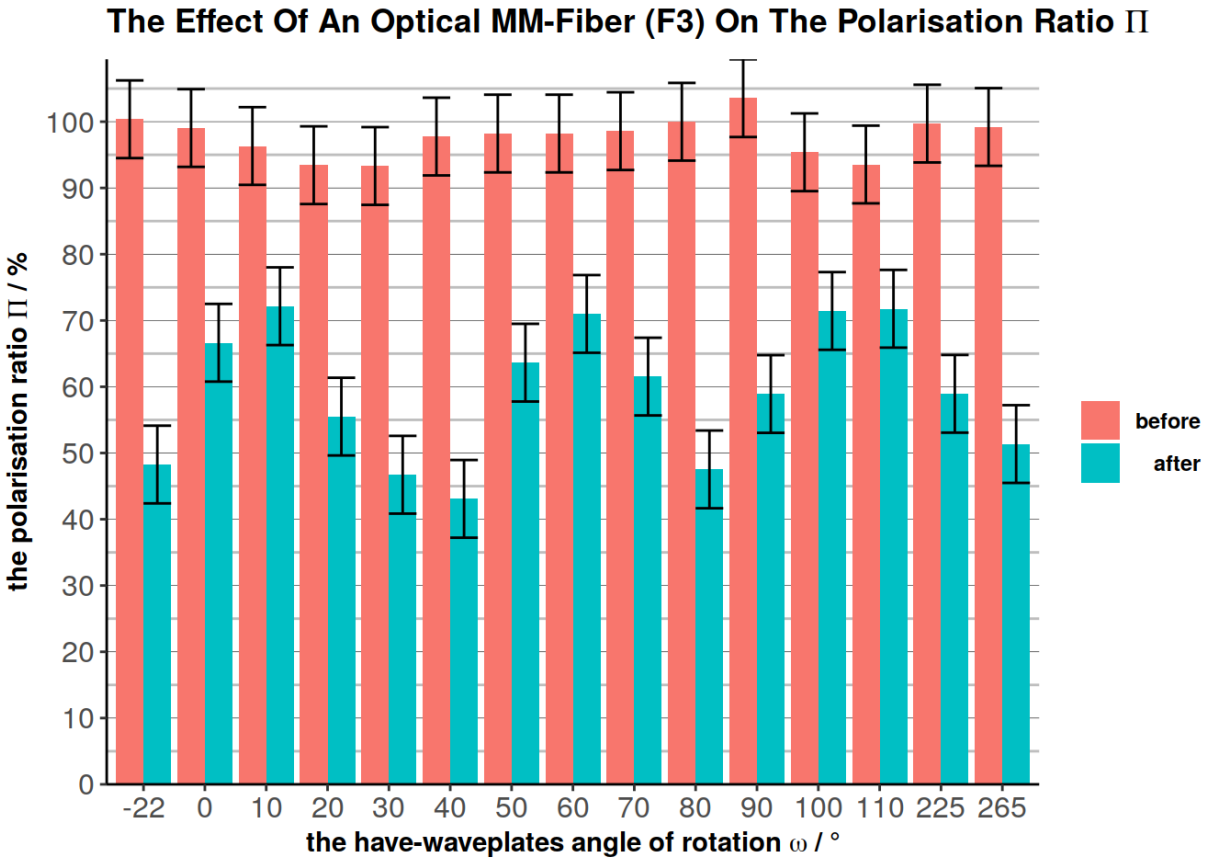
- Die SM-Faser depolarisiert bis zu 50%, kann aber auch die Polarisation erhalten
- Das Depolarisationsverhalten ist stark winkelabhänig
- Das Transmissionsverhalten ist winkelunabhängig
- Es werden ca. 27.5% des Lasers transmittiert
- Die SM-Faser beeinflusst die Polarisationsebene des Lasers
- Die Polarisationsebene wird entlang der 0°-Ebene gespiegelt (Warum?)
- Der Zusammenhang zwischen dem Winkel der Polarisationsebene vor und nach der Faser ist, anders als ertwartet, nicht linear
- Der Einfluss der SM-Faser auf den Polarisationszustand ist nicht vernachlässigbar
- Ähnlich wie PM-Faser: Bei 0° und 90° resultierender Polarisationsgrad sehr gut ($\widehat{=} \; n \cdot 45°$ Wellenplattenrotation). Dazwischen depolarisiert die Faser. Vielleicht verhält sich SM-Faser wie PM-Faser, weil die SM-Faser während des Experiments so gerade gestreckt war wie möglich
- Müllermatrix sagt nur den dritten Stokesparameter korrekt voraus; Polarisationsgrad und ersten zwei Stokesparameter sind falsch
- Warum ist die Müllermatrix falsch? Für F3 hat es funktioniert. Muss das Experiment wiederholt werden? Muss der geschätzte Messfehler berücksichtigt werden?

MM-Faser F3

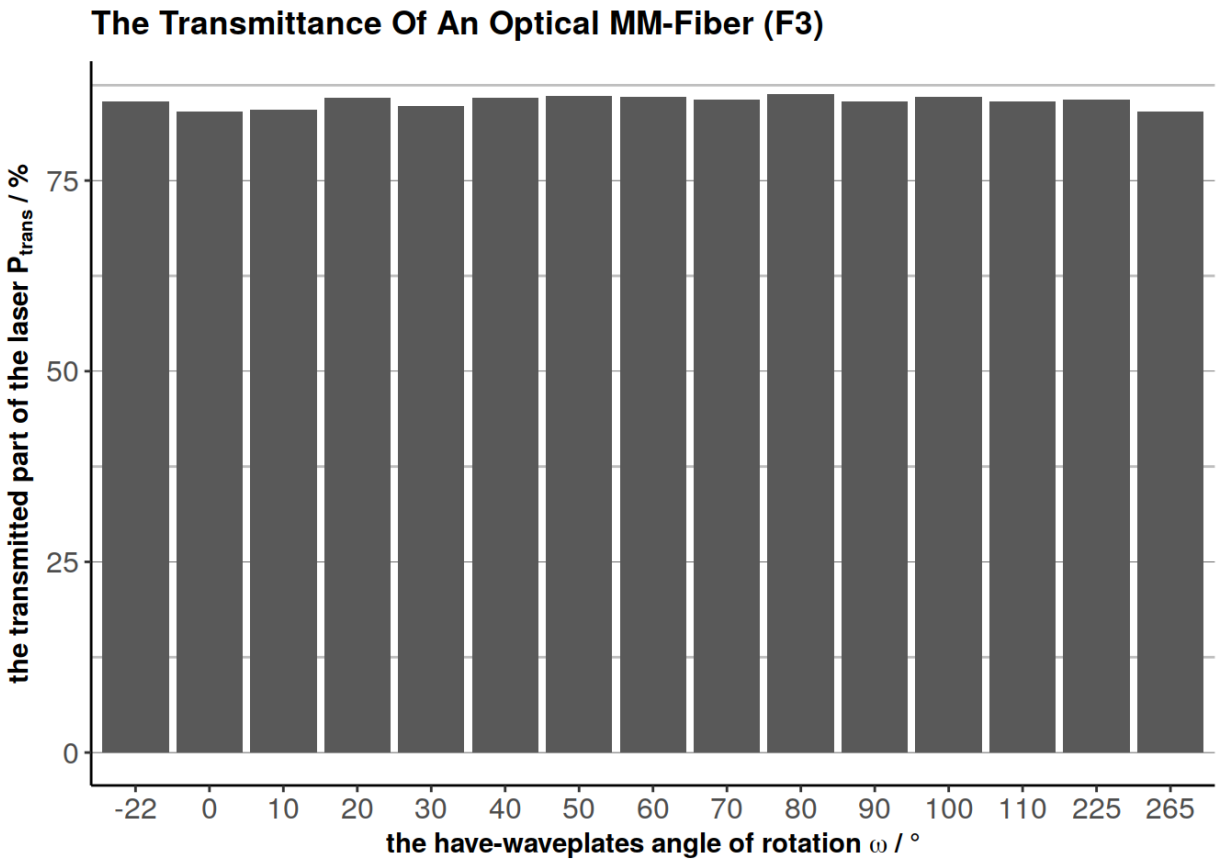
```
# How does the polarisation ratio change relative to the initial polarisation ratio?
plot.polarisation.change(data  = F3.data.stokes,
                         title = expression(bold("The Depolarising Behaviour Of An Optical
MM-Fiber (F3)") )
)
```



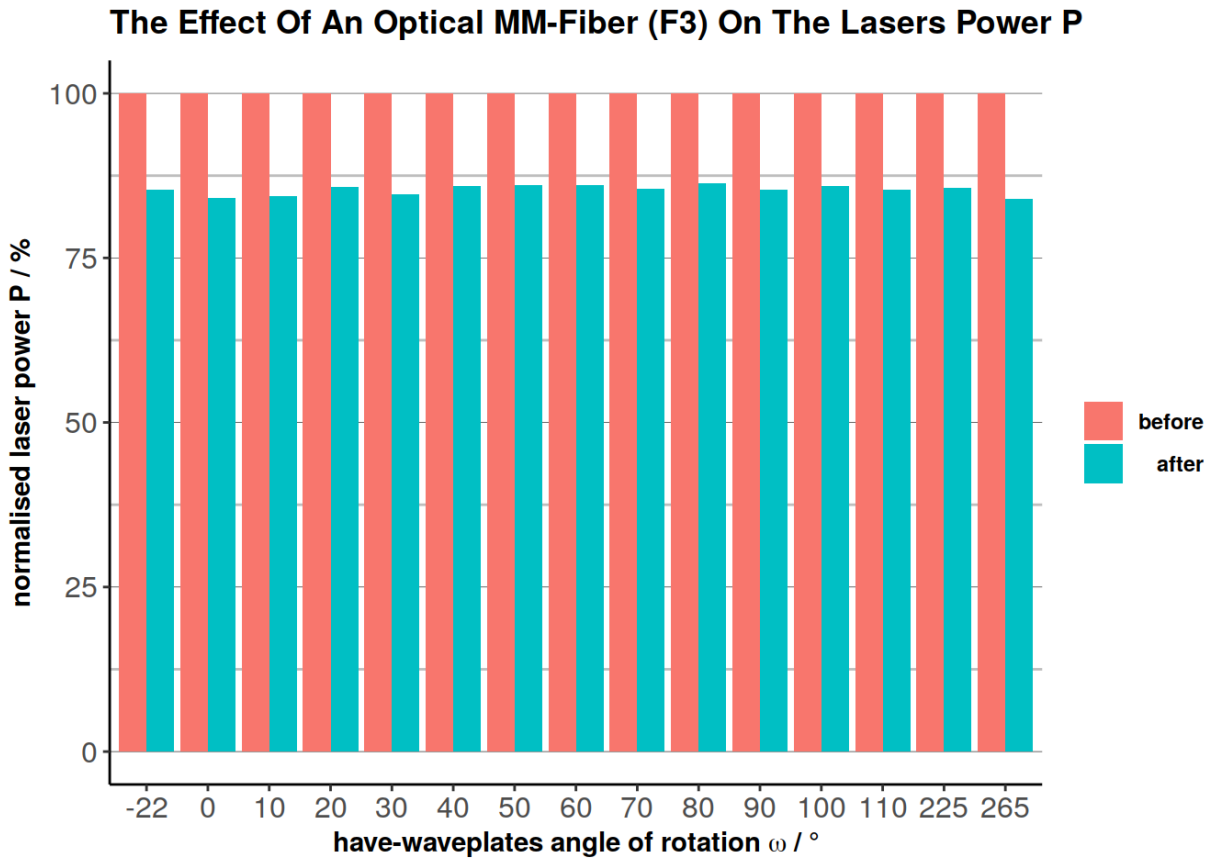

```
# COMPARING POLARISATION RATIOS before and after interacting with the fiber
plot.polarisation(data       = F3.data.stokes,
                  statistics = F3.error.stats,
                  title      = expression(bold("The Effect Of An Optical MM-Fiber (F3) On
The Polarisation Ratio "*Pi))
)
```



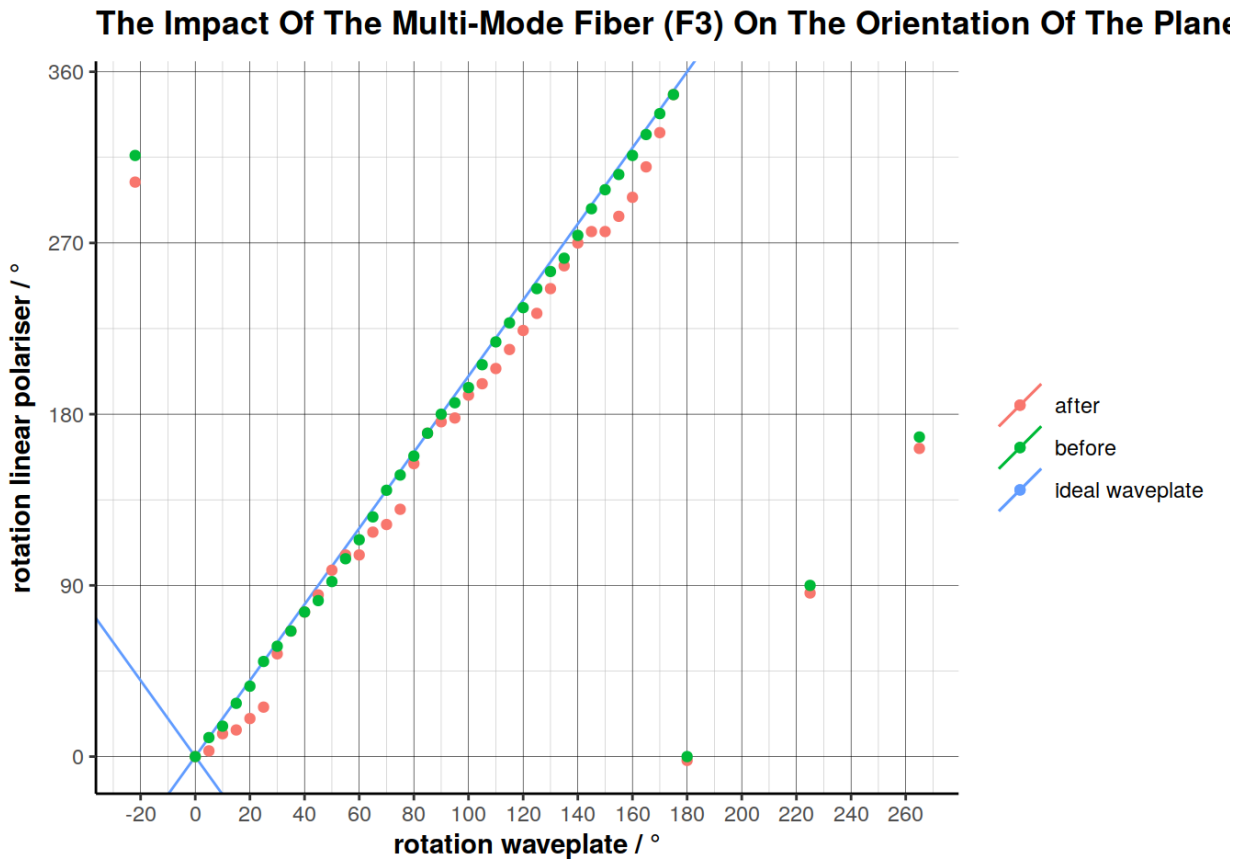
```
# CHANGE in LASER POWER due to optical fiber
plot.intensity.change(data  = F3.data.stokes,
                      title = expression(bold("The Transmittance Of An Optical MM-Fiber (F
3)"))
)
```




```
# COMPARING LASER POWER before and after interacting with the fiber
plot.intensity(data  = F3.data.stokes,
               title = expression(bold("The Effect Of An Optical MM-Fiber (F3) On The Lase
rs Power "*P))
)
```



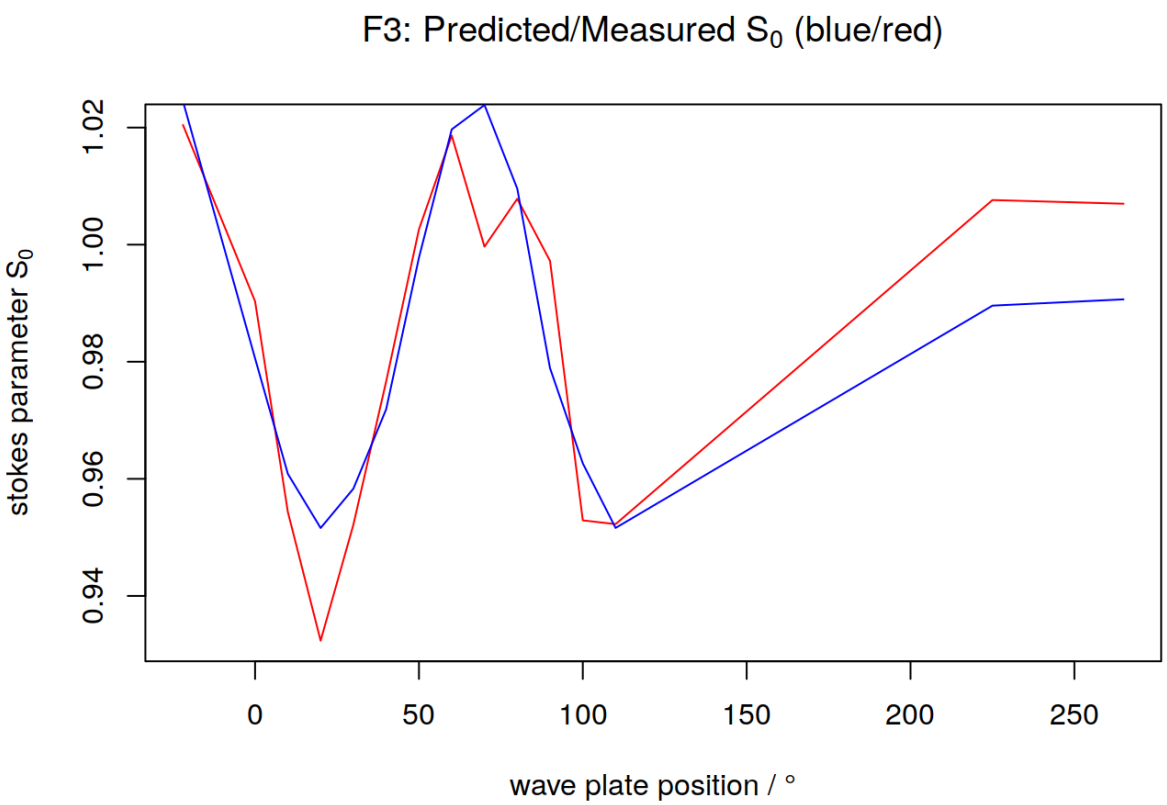

```
# How does the fiber influence the PLANE OF POLARISATIONS ORIENTATION
plot.plane.rotation(F3.rotation.elab,
                    title = expression(bold("The Impact Of The Multi-Mode Fiber (F3) On Th
e Orientation Of The Plane Of Polarisation"))
)
```



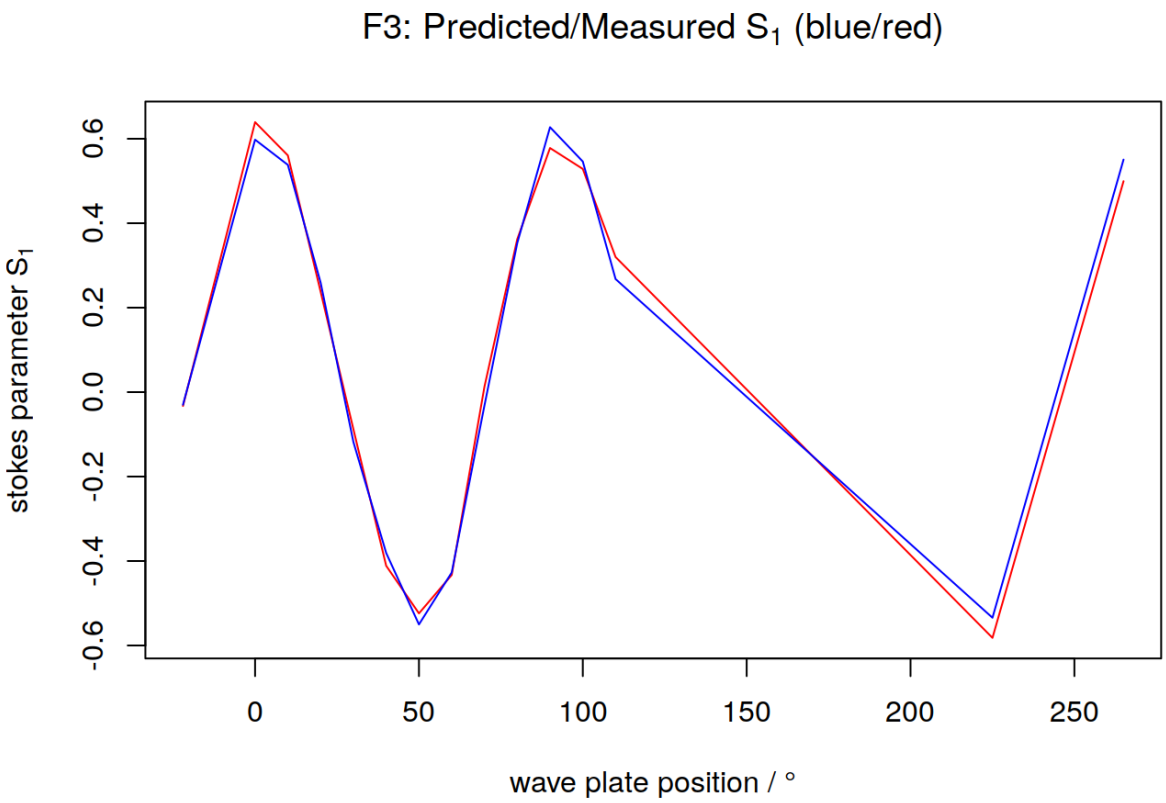
```
# Print calculated mueller matrix
print(F3.muellermatrix)
```

```
##             [,1]         [,2]        [,3] [,4]
## [1,]  0.98734319  0.004664235 -0.03808659    0
## [2,]  0.02519393  0.520569010  0.27315076    0
## [3,] -0.04387837 -0.004022648  0.60062411    0
## [4,]  0.00000000  0.000000000  0.00000000    0
```

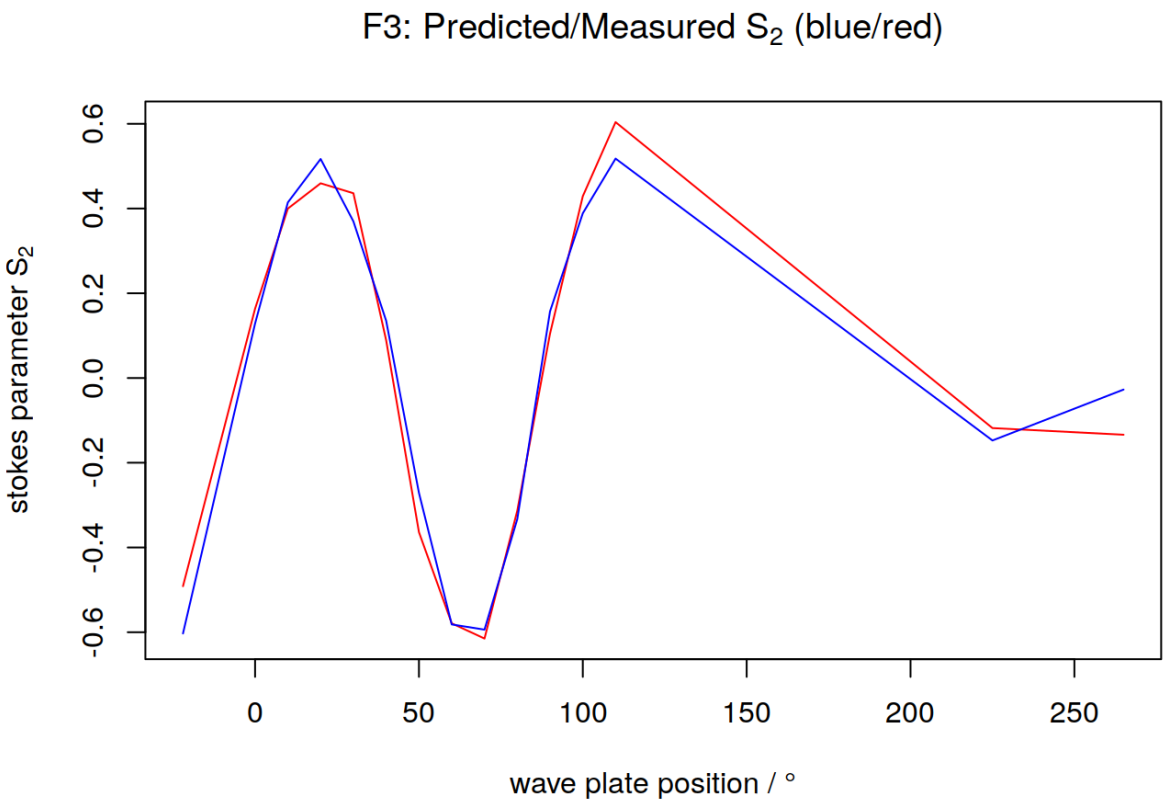
```
# Plot and compare the PREDICTED and MEASURED STOKES parameters
# S0
plot(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST$S0, col="red", type="l",
     main = expression("F3: Predicted/Measured S"[0]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[0]))
lines(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST.PREDICT$S0, col="blue")
```



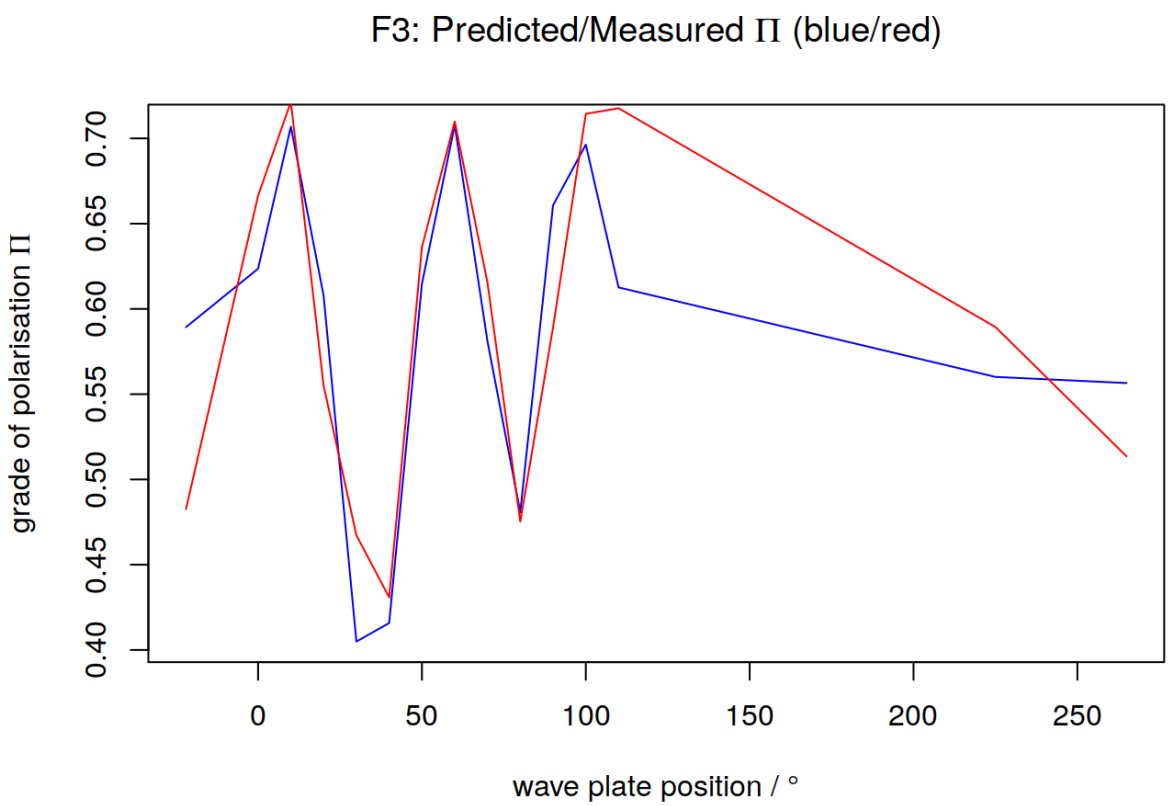

```
# S1
plot(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST$S1, col="red", type="l",
     main = expression("F3: Predicted/Measured S"[1]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[1]))
lines(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST.PREDICT$S1, col="blue")
```




```
# S2
plot(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST$S2, col="red", type="l",
     main = expression("F3: Predicted/Measured S"[2]*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("stokes parameter S"[2]))
lines(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST.PREDICT$S2, col="blue")
```



```
# Polarisation ratio
plot(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST.PREDICT$polarisation, col="blue",
type="l",
     main = expression("F3: Predicted/Measured "*Pi*" (blue/red)"),
     xlab = "wave plate position / °",
     ylab = expression("grade of polarisation "*Pi))
lines(x = F3.data.stokes$POST$W, y = F3.data.stokes$POST$polarisation, col="red")
```



Fazit:

- MM-Faser reduzert Polarisationsgrad um ~22° bis ~37°
- Depolarisierungseigenschaften periodisch. Maximum alle 50°. Komische Periode.
- Charakterisierung der Depolarisationseigenschaften schwierig, weil zwischen MM-Faser und Probe das Mikroskop ist
- Transmissionsverhalten sehr gut und winkelunabhängig
- Transmission ~85%
- Der Zusammenhang zwischen dem Winkel der Polarisationsebene vor und nach der Faser ist, anders als ertwartet, nicht linear
- Der Einfluss der MM-Faser auf den Polarisationszustand ist nicht vernachlässigbar
- Müllermatrix sagt den Stokesvektor und den Polarisationsgrad gut voraus. Ist es genau genug, um die Matrix in PolaRam zu benutzen?

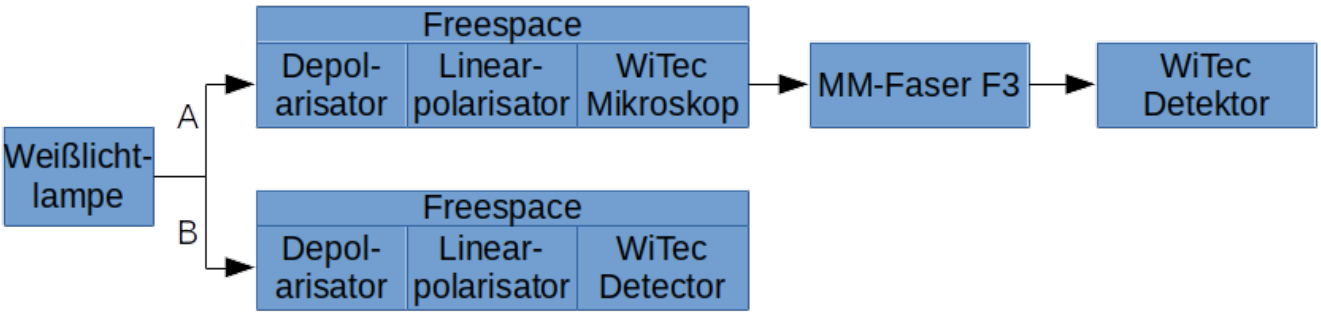
WiTec-Detektor

Experiment

Es wird ein Experiment mit zwei verschiedenen Aufbauten durchgeführt:

- Polarisationsempfindlichkeit des Detektors: Unterscheiden sich die Weißlichtspektren von unterschiedlich linear polarisiertem Lichtquellen?

Polarisationsempfindlichkeit:



Aufbau Polarisationsempfindlichkeit des WiTec-Detektors (A: Mit Mikroskop, B: Ohne Mikroskop)

Durchführung:

- Für verschiedene Positionen des Linearpolarisators werden Ramanspektren aufgenommen
- Wichtig: Die Position des Linearpolarisators im Freespace ist nicht mit der Position des selben Polarisators in der Fiberbench vergleichbar!

Auswertung

- Definition von Funktionen zum Plotten und organisieren der Daten

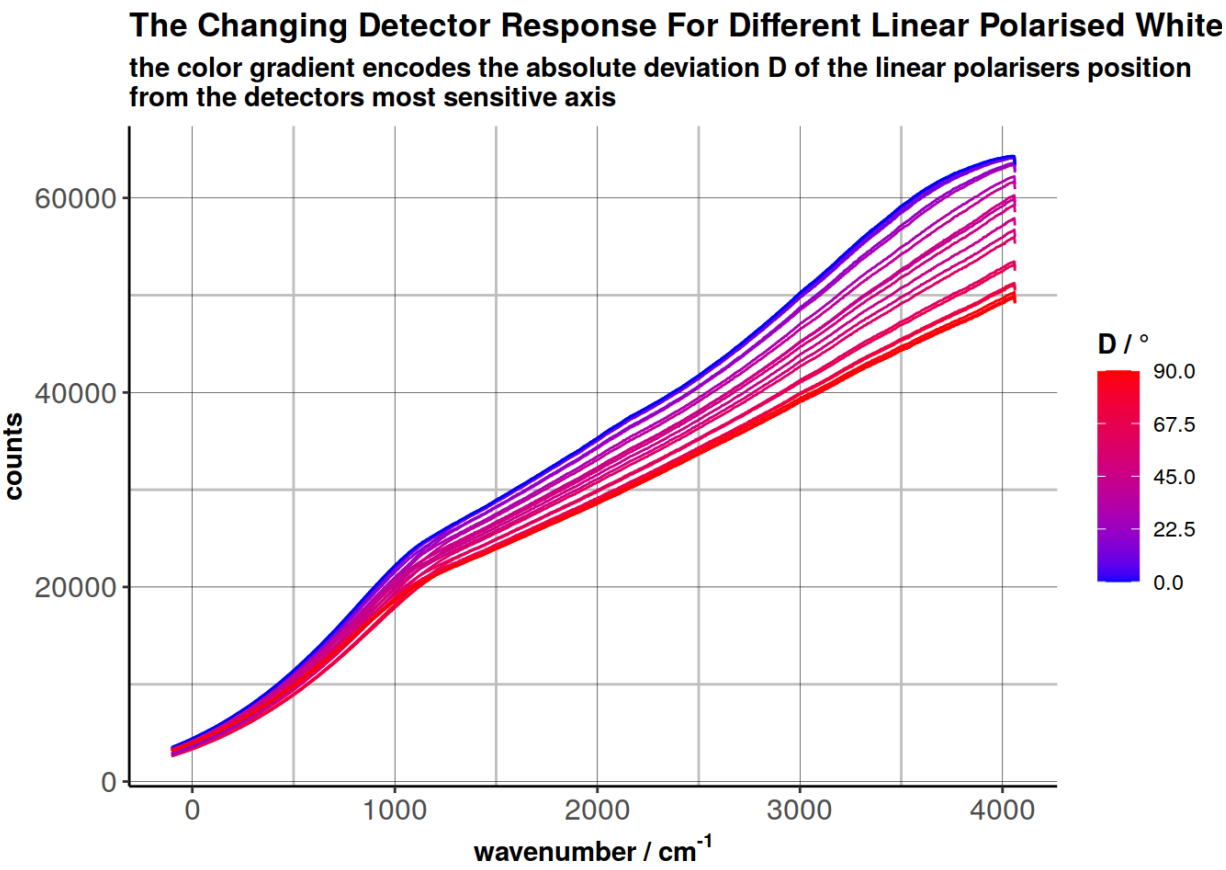
```
# CONVERT TIME SERIES OF SPECTRA INTO EASY PLOTABLE DATA.FRAME
# This function turns a data.frame with multiple spectra organised in multiple columns int
o a data.frames with all spectra staked in the same columns
# Makes plotting with ggplot easier
makeSpectraPlotable <- function(spectra, colorFunc=function(x) return(x)) {
  # PARAMETERS
  # spectra : the return value of parseTimeSeries.elab, contains a time series with multip
le spectra
  # colorFunc : a custom function applied to the time variable (in this case the position
of the linear polariser) to tweak the color scale
  # RETURN VALUE
  # A data.frame with all spectra stacked on top of each other. data.frame has the column
s: wavenumber, signal, P (linear polarisers rotation), color

  # Extract the position of the linear polariser from the column names and repeat the valu
e to match the length of the corresponding spectrum
  # Used for colouring and grouping the data correctly when plotting with ggplot
  polariser <- lapply(seq_along(spectra[,-1])+1, function(index) {
    rep( colnames(spectra)[index], length.out=length(spectra[,index]) )
  } ) %>% unlist %>% as.numeric

  # Create a dataframe with all spectra stacked on top of each other, instead of every spe
ctrum in an own column
  data.frame(wavenumber   = spectra$wavenumber,
             signal       = unlist(spectra[,-1]) %>% unname,
             P            = polariser,
             # Hand the polariser position to a custom function to tweak the color scale
             color        = colorFunc(polariser)
  )
}



#
# PLOTING FUNCTIONS
#
# Plot all measured WHITE LIGHT SPECTRA in one plot and color code them according to the r
otation of the linear polariser
# The color should show the absolute DEVIATION of the lasers plane of polarisation FROM TH
E DETECTORS MOST SENSITIVE AXIS
plot.detector.whitelamp <- function(data,
                                    title = "The Changing Detector Response For Different
Linear Polarised White Light <Of Your Equipment>"
                                   ) {
  # PARAMETERS
  # data : plotable time series of spectra. Use the return value of RHotStuff::parseTimeSe
ries.elab() %>% makeSpectraPlotable()
  # title : Some descriptive title

  ggplot( data = data,
          mapping = aes(x = wavenumber, y = signal, group = P, color = color)
         ) +
    scale_color_gradient(low    = "blue",
                         high   = "red",
                         breaks = seq(from=0, to=90, by=22.5)
                        ) +
    theme_hot() +
    labs(title = title,
         y = "counts",
```

```
        x = expression(bold("wavenumber / cm"^"-1")),
        subtitle = "the color gradient encodes the absolute deviation D of the linear pol
arisers position \nfrom the detectors most sensitive axis",
        color = "D / °") +
    geom_line()

}

# Plot WHITE LAMP SPECTRA in one 3d plot as 3D SURFACE (single picture)
plot.detector.allSpectra <- function(data,
                                     title = expression(bold("The White Lamp Raman Spectra
For Different Polarised Light")),
                                     color.resolution = 100,
                                     color.ramp = c("blue", "red"),
                                     theta = 270,
                                     phi = 20,
                                     grid.resolution.X = 20,
                                     grid.resolution.Y = 2

) {
  # Seperate wavenumber axis, polariser position and spectra
  PlotMat <- as.matrix(data[, -1])
  wavenumber <- data$wavenumber
  polariser <- as.numeric( colnames(PlotMat) )

  # Create a grid for plotting
  grid <- list(ordinate = wavenumber, abcissa = polariser)
  grid.surface <- make.surface.grid(grid)

  # Create a 3d plottable surface
  surface <- as.surface(grid.surface, PlotMat)

  # Create color palette
  col.Palette <- colorRampPalette(color.ramp)(color.resolution)
  # Calculate Color of the surface according to the z-value of the corresponding point
  zfacet <- PlotMat[-1, -1] + PlotMat[-1, -ncol(PlotMat)] + PlotMat[-nrow(PlotMat), -1] +
PlotMat[-nrow(PlotMat), -ncol(PlotMat)]
  facetcol <- cut(zfacet, color.resolution)
  plotCol <- persp(surface, theta=theta, phi=phi)

  # Create the plor
  plot.surface(surface, type="p", theta=theta, border=NA, phi=phi,
               xlab = "wavenumber / cm^-1",
               ylab = "wave plate position / °",
               zlab = "detector signal / counts",
               main = title)

  # Add grid lines
  # Get the position of the gridlines
  select.X <- seq(1,length(grid[[1]]), by=grid.resolution.X)
  select.Y <- seq(1,length(grid[[2]]), by=grid.resolution.Y)
  xGrid <- grid[[1]][select.X]
  yGrid <- grid[[2]][select.Y]

  # Draw the gridlines
  for(i in select.X) lines(trans3d(x=rep(grid[[1]][i],ncol(PlotMat)),
                                   y=grid[[2]],
                                   z=PlotMat[i,], pmat=plotCol))
  for(i in select.Y) lines(trans3d(x=grid[[1]],
```

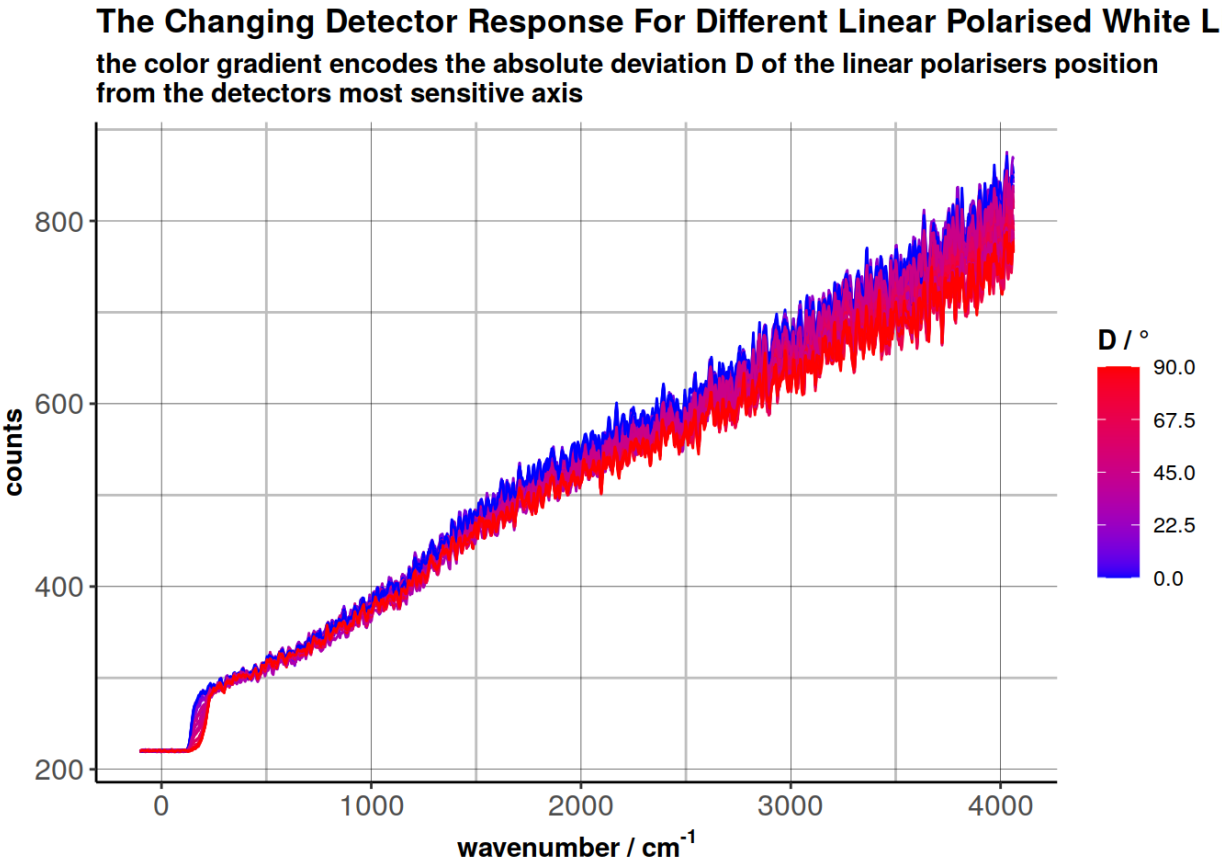
```
                                y=rep(grid[[2]][i],nrow(PlotMat)),
                                z=PlotMat[,i], pmat=plotCol))
}
```

- Spektren herunterladen
- .txt-Dateien mit den Ramanspektren auslesen und alle Spektren in einem data.frame vereinigen
- Die Spektren werden coloriert, wenn sie in einem Plot überlagert werden
- Positionen des Linearpolarisators p, die die selbe absolute Auslenkung aus der 90°/270°-Achse α haben, werden gleich coloriert: $\alpha = |(p \bmod \pi) - \frac{\pi}{2}|$

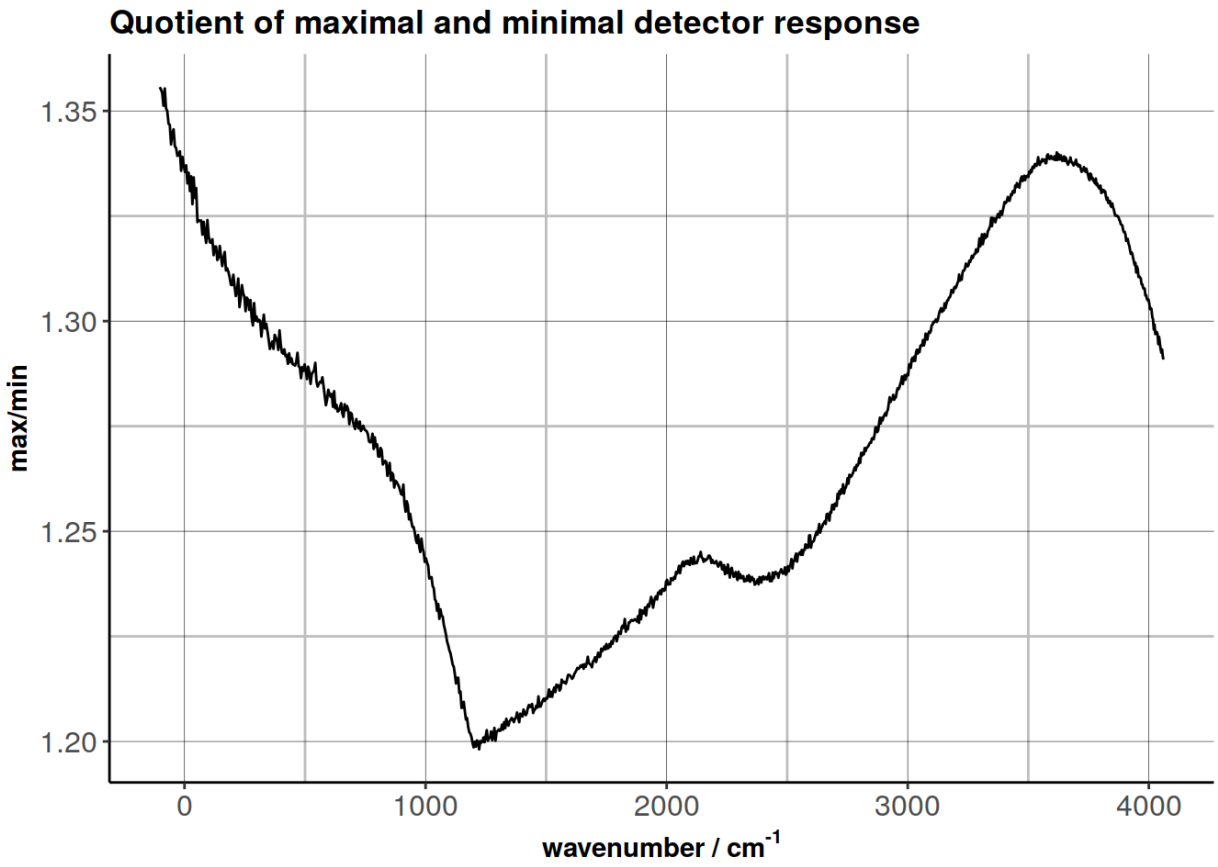
```
# Fetch experimental data from elabFTW
detector.spectra <- GET.elabftw.bycaption(76, header=T, outputHTTP=T) %>% parseTimeSeries.
elab(., header=F, sep="")


#
# PLOT THAT SHIT
#

# Plot the white lamp spectra for the detector without the microscope
plot.detector.whitelamp(data=makeSpectraPlotable(detector.spectra[[2]],
                                                 colorFunc=function(polariserRotation) {mo
d(polariserRotation, 180) %>% `-`(.,90) %>% abs(.)} ),
                        title="The Changing Detector Response For Different Linear Polaris
ed White Light Of The WiTecs Detector")
```



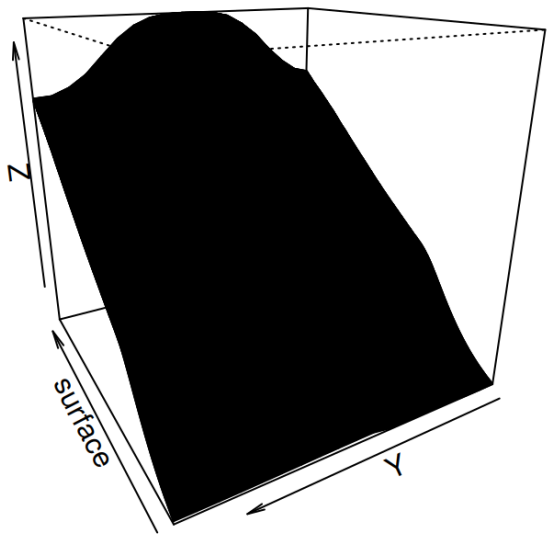
```
# Plot the white lamp spectra for the detector with the microscope
plot.detector.whitelamp(data=makeSpectraPlotable(detector.spectra[[1]],
                                                 colorFunc=function(polariserRotation) {mo
d(polariserRotation, 180) %>% `-`(.,90) %>% abs(.)} ),
                        title="The Changing Detector Response For Different Linear Polaris
ed White Light Of The WiTecs Detector And Microscope")
```



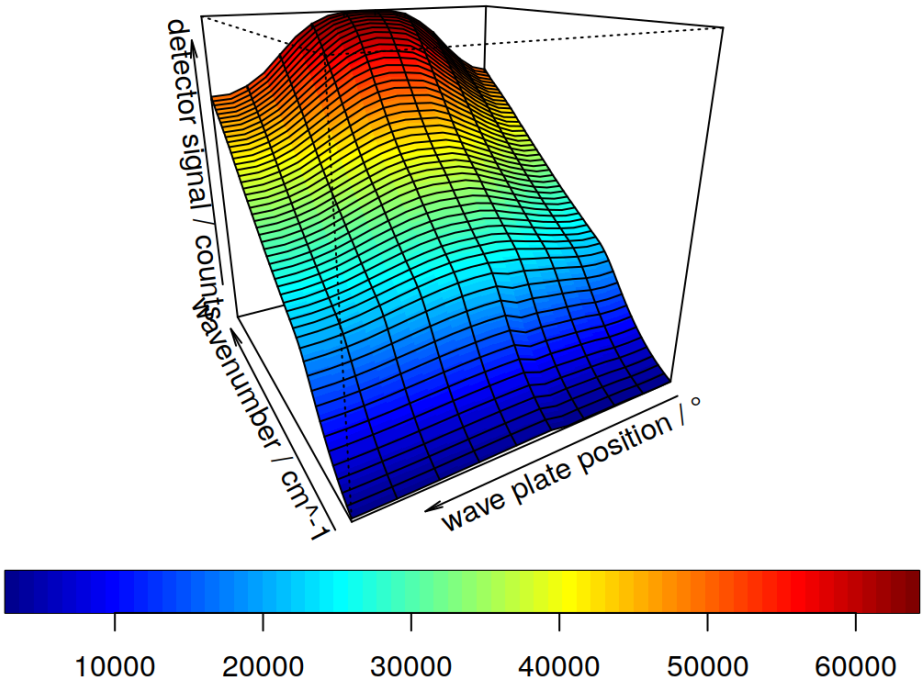

```
# How does the sensitivity of the detector change with the wavenumber (quotient method)
ggplot( data = data.frame(wavenumber = detector.spectra[[2]]$wavenumber,
                          quotient   = apply(detector.spectra[[2]][,-1], 1, function(slic
e) { max(slice)/min(slice) })
                          ),
        mapping = aes(x=wavenumber, y=quotient) ) +
  theme_hot() +
  labs(title = "Quotient of maximal and minimal detector response",
       x = expression(bold("wavenumber / cm"^"-1")),
       y = "max/min") +
  geom_line()
```




```
# Plot the WHITE LAMP SPECTRA in one 3d plot as 3D SURFACE
plot.detector.allSpectra(detector.spectra[[2]][,-c(21:24)], theta=240)
```



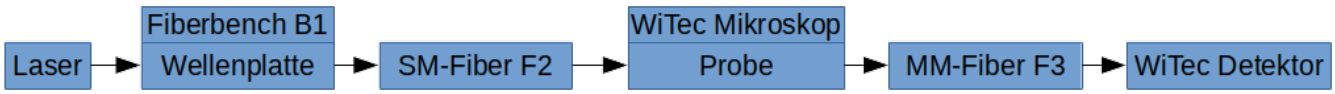
The White Lamp Raman Spectra For Different Polarised Light



Fazit:

- Die Messung mit dem Mirkoskop ist sehr verrauscht. Die Weißlichtlampe ist nicht hell genug dafür. Mit langer Integrationszeit wiederholen.
- Weißlichtspektren zeigen Abhängigkeit von Lichtpolarisation
- Nicht vergessen: Die Linearpolarisatorposition kann nicht vom Freespace auf die Fiberbenches übertragen werden!
- Für jede Wellenzahl gibt es je eine Polarisation, die das gemessene Signal maximiert bzw. minimiert
- Der Detektor ist entlang der 0°-Achse empfindlicher als entlang der 90°-Achse
- Der Quotient aus max/min beschreibt wie empfindlich der Detektor entlang der 0°-Achse im Vergleich zur 90°-Achse ist
- Die Form des Graphen ist komisch. Komische Form könnte am russischen Aufbau liegen. Messung wiederholen, um Reproduzierbarkeit zu überprüfen
- Die Detekorempfindlichkeit schwankt zwischen 1.20:1 und 1.35:1 ($I_{0^\circ} : I_{90^\circ}$ bzw. $I_x : I_y$)

Polarisaionsabhängige Ramanspektroskopie am WiTec



Aufbau Messung von Ramanspektren mit spezifischer Laserpolarisation

Durchführung:

- Für verschiedene Positionen der Wellenplatte wird ein Ramanspektrum einer Probe aufgenommen
- Für jedes Spektrum wird die Leistung des Lasers am Mikroskop gemessen

Ramanspektren von Tetrachlormethan

Auswertung:

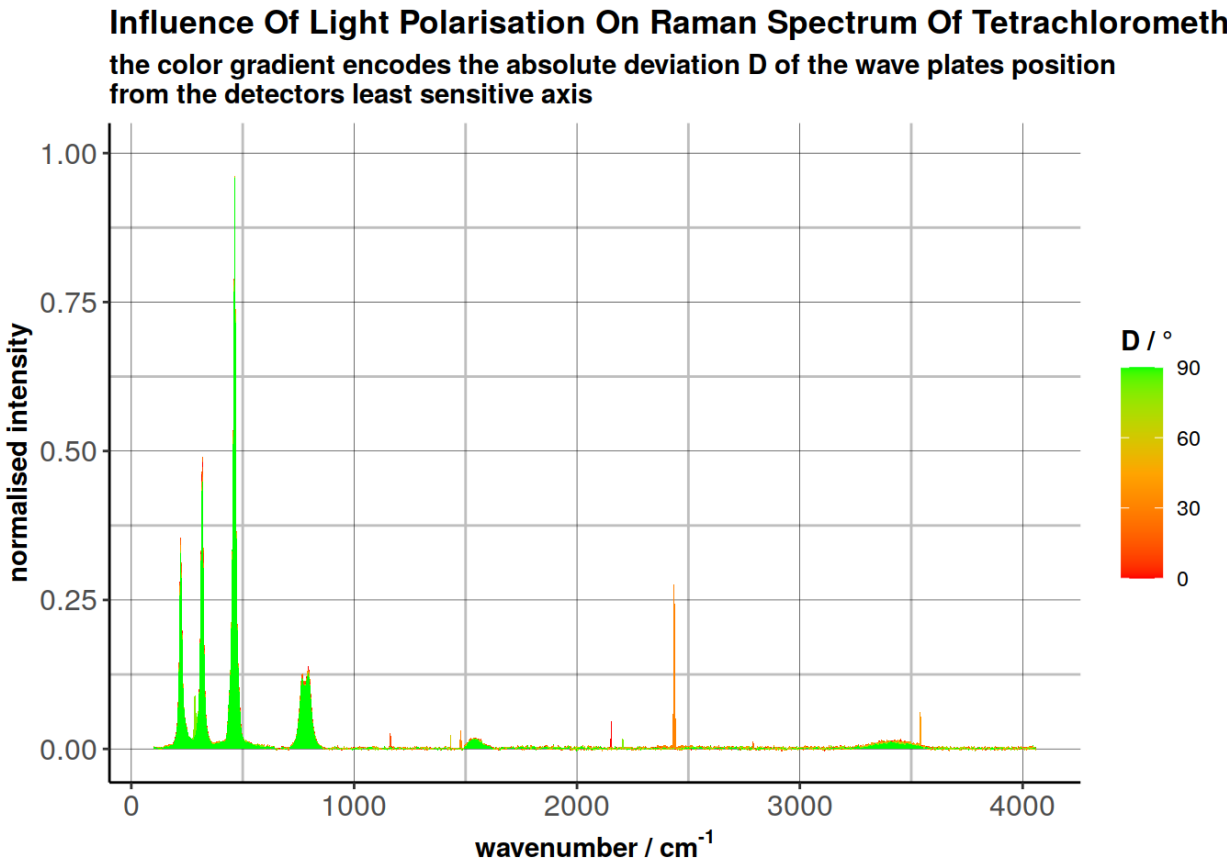
- Messdaten herunterladen (alle Spektren in einem data.frame, erste Spalte Wellenzahlachse, restliche Spalten Ramanspektren, Spaltennamen Positionen der Wellenplatte)
- Preprocess: “Background Correction” mit fillPeaks-Methode des baseline-Pakets, Normalisieren der Spektren mit der Höhe des größten Peaks
- Finden aller Peaks im Ramanspektrum
- Änderung der Peakhöhen für verschiedene Wellenplattenpositionen in neue Matrix schreiben
- Vergleichen der Empfindlichkeit des Detektors für Licht, dass entlang der optischen Achsen des Detektors polarisiert ist
- Vergleich der Empfindlichkeit Q als Quotient der maximalen Peakhöhe h_{max} und minimalen Peakhöhe h_{min}: $Q = \frac{h_{max}}{h_{min}}$

```
# FETCH DATA
tetra.spectra <- GET.elabftw.bycaption(79, header=T, outputHTTP=T) %>%
                  parseTimeSeries.elab(., col.spectra=3, sep="") %>% .[[1]]

# PREPROCESS
# Statistical Background Correction
# Peaks no longer available
# tetra.spectra[,-1] <- sapply(tetra.spectra[,-1], function(spec) spec-Peaks::SpectrumBack
ground(spec))
tetra.spectra[,-1] <- t( as.matrix(tetra.spectra[,-1]) ) %>%
                        baseline::baseline(., method="fillPeaks", lambda=1, it=10, hwi=50,
int=2000) %>%
                        baseline::getCorrected(.) %>% t(.)
# Normalise each spectra by its highest peak
tetra.spectra[,-1] <- sapply(tetra.spectra[,-1], function(spec) spec/max(spec))


# FIND THE LOCATION OF THE PEAKS
# Guess the general area (wavenumbers) where the peak lays in
tetra.peakMargins   <- list( c(100, 250),
                             c(250, 340),
                             c(340, 580),
                             c(580, 770),
                             c(770, 890) )
# Find the maximum in each guessed area
tetra.peakLocations <- sapply(tetra.peakMargins, function(margins) {
  # Guess the broad location of the peak
  selectGeneralPeakLocation <- which(tetra.spectra$wavenumber > margins[1] & tetra.spectr
a$wavenumber < margins[2])
  # Find the maximum of the guessed area and returns its location
  peak.value <- max( tetra.spectra$`0`[selectGeneralPeakLocation] )
  peak.location <- tetra.spectra$wavenumber[tetra.spectra$`0` == peak.value]
  return(peak.location)
})


# EXTRACT THE PEAK HEIGHTS FOR DIFFERENT LASER POLARISATIONS
# Create empty matrix
tetra.peakChange <- matrix( NA, nrow = ncol(tetra.spectra[,-1]), ncol = length(tetra.peakL
ocations)+1 )
# Add column with the wave plate position
tetra.peakChange[,1] <- colnames(tetra.spectra[,-1]) %>% as.numeric
# Loop over all peaks and and add their heights to the matrix
for (index in seq_along(tetra.peakLocations)) {
  # Get the wavenumber of the peak
  peak <- tetra.peakLocations[index]
  # Add the corresponding row of the spectra table to the matrix
  tetra.peakChange[,index+1] <- tetra.spectra[ which(tetra.spectra$wavenumber==peak),-1 ]
%>% unlist
}
# Add describtive column names
colnames(tetra.peakChange) <- c("waveplate", tetra.peakLocations)
# Translate matrix to data.frame
tetra.peakChange <- as.data.frame(tetra.peakChange)


# COMPUTE DETECTORS SENSITIBLITY FOR LIGHT POLARISED ALONG DIFFERENT OPTICAL AXIS
tetra.sensibility <- sapply(tetra.peakChange[,-1], function(peakheight) max(peakheight)/mi
n(peakheight))
```

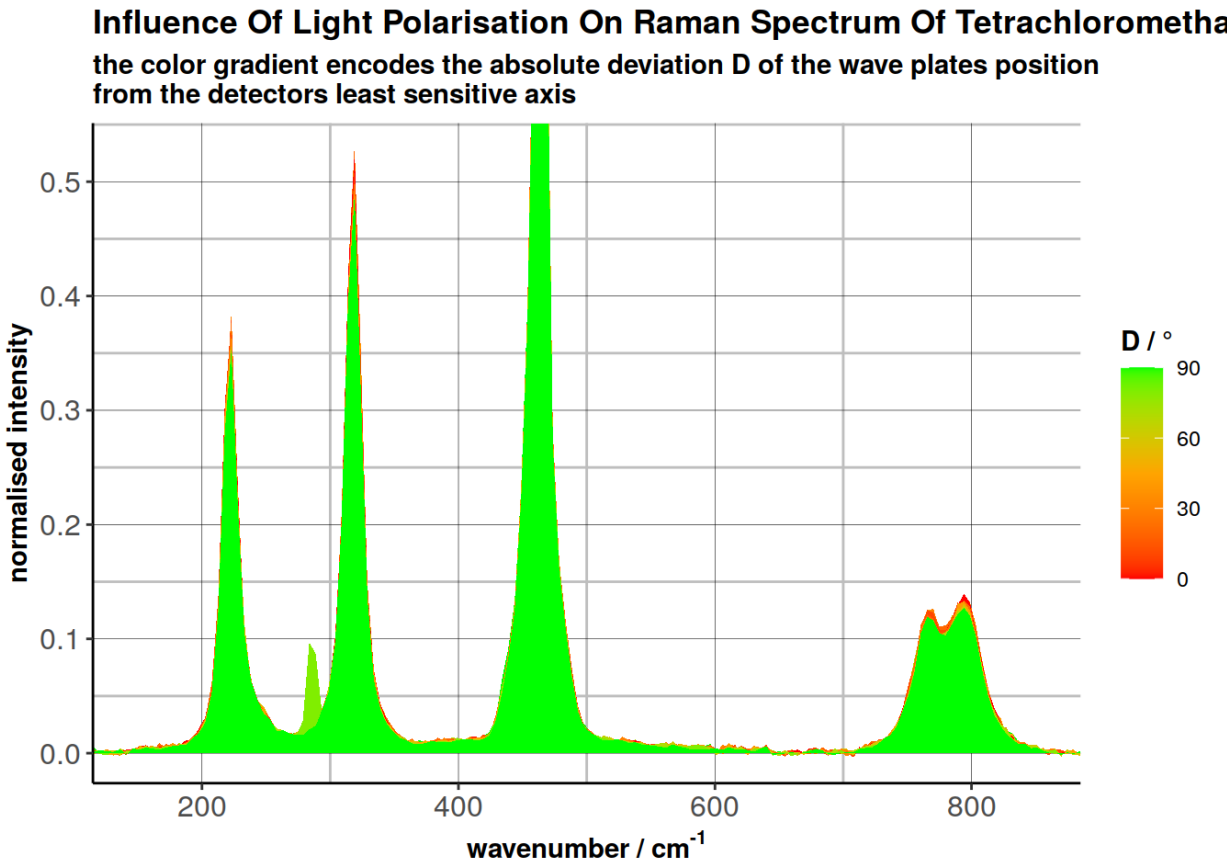
```
tetra.sensibility <- data.frame(wavenumber = names(tetra.sensibility) %>% as.numeric,
                                quotient   = tetra.sensibility %>% unname)


# HOW ARE THE OPTICAL AXIS OF THE DETECTOR ALIGNED?
# The wave plates position of the maximal detector response
# Loop over all peaks and compare the height between the different spectra
# sapply(seq_along(tetra.peakChange[,-1]), function(index) {
#   # The maximal height of the current peak
#   peakHeight <- max(tetra.peakChange[,index+1])
#   # Get the index of the maximum
#   select <- which(peakHeight==tetra.peakChange[,index+1])
#   # If there are multiple maxima its probably the peak I normalised with
#   # Just return NA in this case
#   if (length(select)>1) return(NA)
#   # Return the wave plates position
#   return( tetra.peakChange[ select,1 ] )
# })
```

- Die Spektren werden coloriert, wenn sie in einem Plot überlagert werden
- Positionen der Wellenplatte w, die die selbe absolute Auslenkung der Polarisationsebene aus der 90°/270°-Achse α haben, werden gleich coloriert: $\alpha = |(2w \bmod \pi) - \frac{\pi}{2}|$

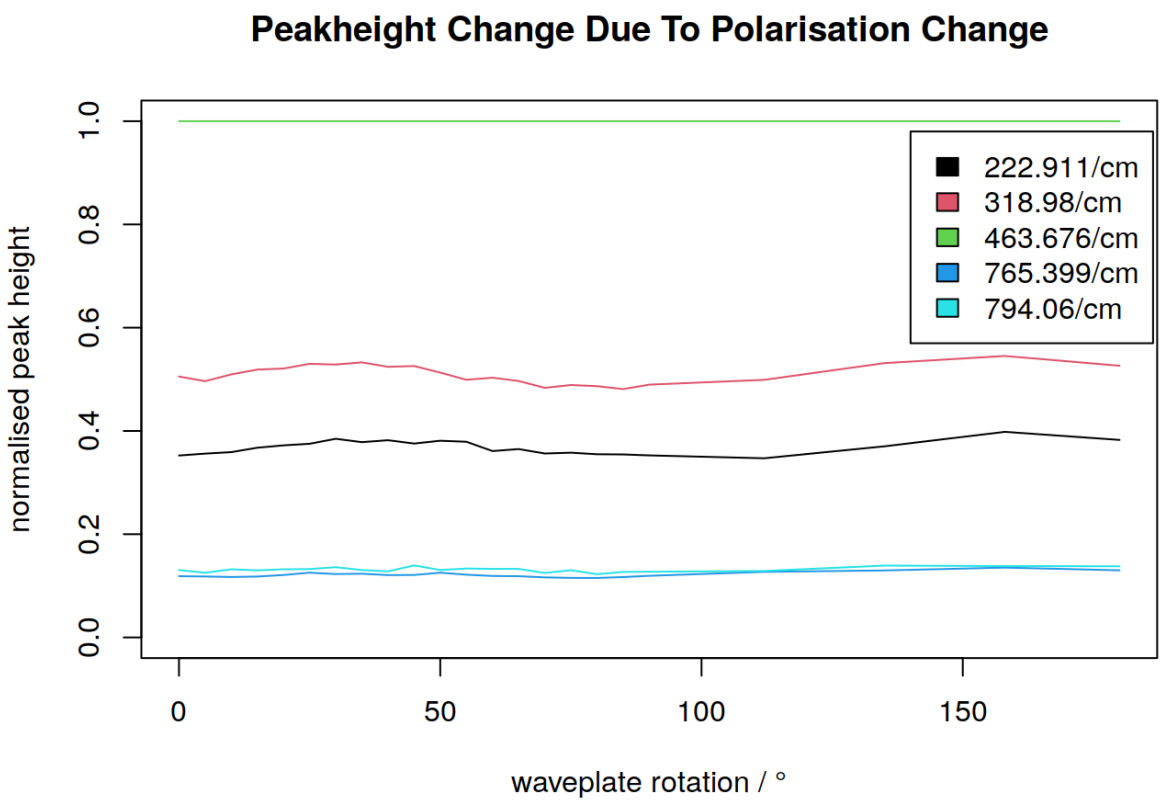
```
# Plot ALL SPECTRA OVERLAYERD as 2d plot
tetra.plot.allSpetra <- ggplot( data = makeSpectraPlotable(tetra.spectra[tetra.spectra$wav
enumber>100,-c(21:24)],
                                                           colorFunc=function(waveplateRot
ation)
                                                           { mod(waveplateRotation*2, 180)
%>% `-`(., 90) %>% abs } ),
                                mapping = aes(x = wavenumber, ymax = signal, ymin=0, group
= P, fill = color) ) +
   scale_fill_gradientn(colors = c("red", "orange", "green"),
                        breaks = seq(from=0, to=90, length.out=4) ) +
  theme_hot() +
  labs(title = "Influence Of Light Polarisation On Raman Spectrum Of Tetrachloromethane",
       y = "normalised intensity",
       x = expression(bold("wavenumber / cm"^"-1")),
       subtitle = "the color gradient encodes the absolute deviation D of the wave plates
position \nfrom the detectors least sensitive axis",
       fill = "D / °") +
   geom_ribbon()
# Plot all wavenumbers
tetra.plot.allSpetra
```



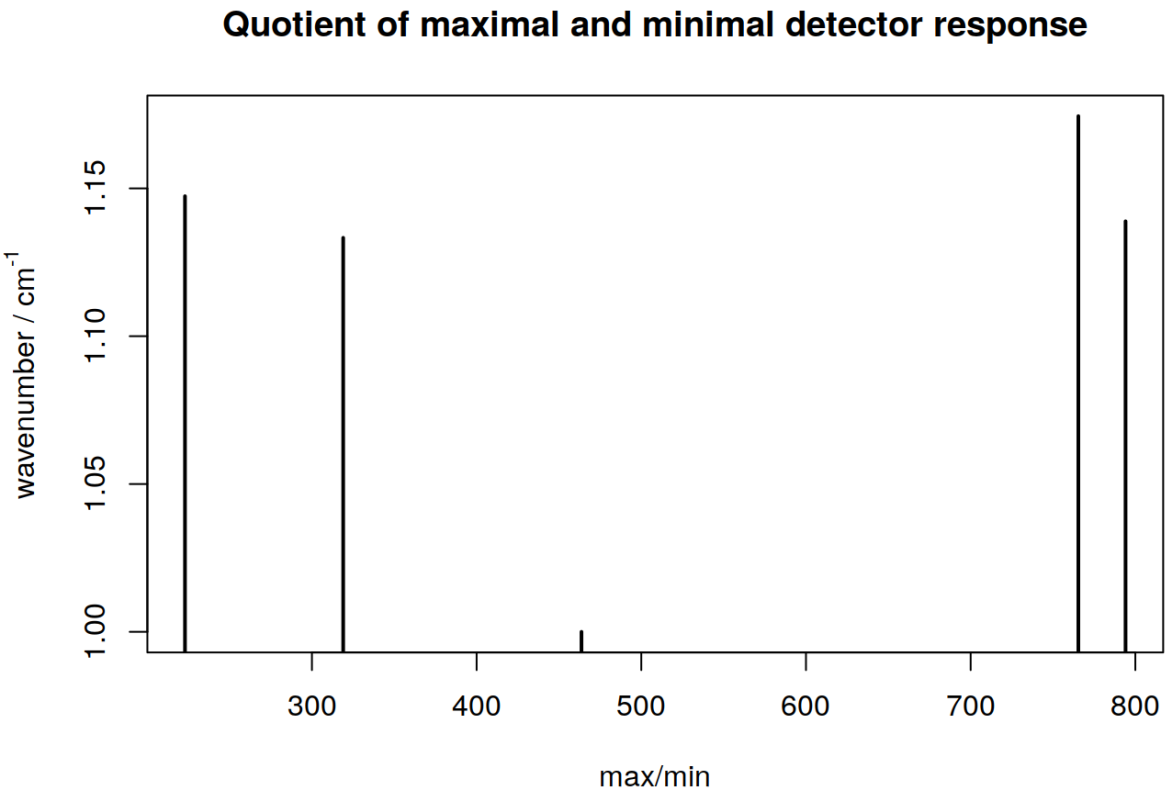

```
# Show just the interesting part
tetra.plot.allSpetra + coord_cartesian(xlim = c(150, 850), ylim = c(0,0.525))
```



```
# Plot the HEIGHT OF PEAKS against the wave plates position
plot(x=tetra.peakChange$waveplate, y=tetra.peakChange[,2], type="n", ylim=c(0,1),
     main = "Peakheight Change Due To Polarisation Change",
     xlab = "waveplate rotation / °",
     ylab = "normalised peak height")
for (index in seq_along(tetra.peakChange[,-1])) lines(x=tetra.peakChange$waveplate, y=tetr
a.peakChange[,index+1], col=index)
legend( x = 140, y = 0.98,
        legend = paste0(colnames(tetra.peakChange[,-1]), "/cm"),
        fill = 1:ncol(tetra.peakChange[,-1]) )
```




```
# Plot the quotient of the DETECTOR RESPONSE along its optical axis
plot(tetra.sensibility, type="h", lwd=2,
     main = "Quotient of maximal and minimal detector response",
     xlab = "max/min",
     ylab = expression("wavenumber / cm"^"-1") )
```



Fazit:

- Peakhöhe des Ramanspektrums hängt von der Polarisation des Lasers ab
- Peakhöhenverhältnisse hängen von der Polarisation des Lasers ab
- Detektorempfindlichkeit ~1.15:1
- Empfindlichkeit geringer als bei der Charakterisierung des Detektors ermittelt
- Empfindlichkeit hängt viel weniger von der Wellenzahl ab als bei der Charakterisierung des Detektors ermittelt
- Andere Detektorempfindlichkeit liegt wahrscheinlich daran, dass bei der Charakterisierung das Mikroskop nicht berücksichtigt wurde
- Detektorempfindlichkeit für alle Peaks unterschiedlich
- Detektorempfindlichkeit nicht für höchsten Peak bestimmbar, weil auf den höchsten Peak normiert wird. Andere Normierungsmethode wählen?