AFModulus_Flex

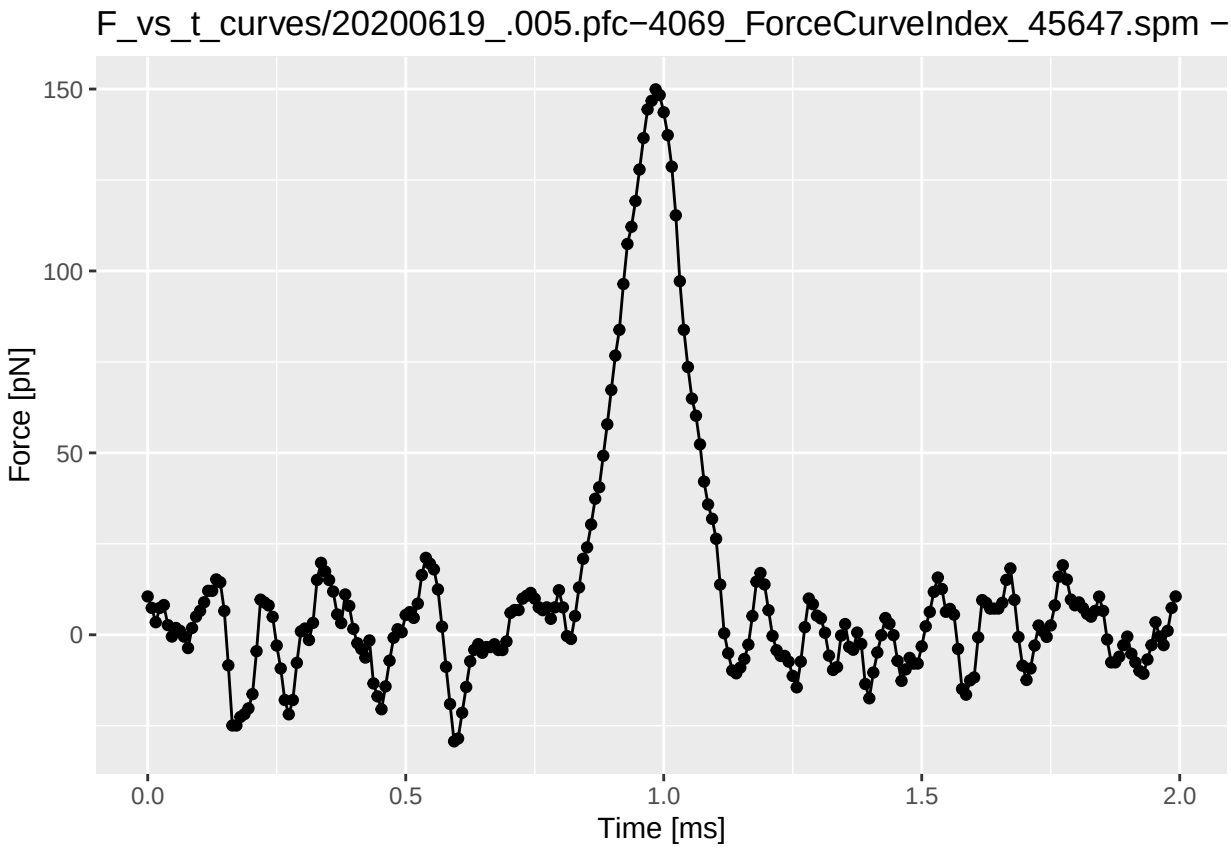
Selen Manioglu, Astrid Stubbusch, Audrey Yeo

1/11/2021

Atomic Force Microscopy - From Topology to Stiffness (Modulus)

The git page of this project can be found here https://github.com/audreyyeoCH/AFModulus_Flex.

Import of AFM curves (force vs. separation distance) for each pixel



Plot 1 curve

We will use the force signal between time 0.0 - 0.5 ms as well as 1.5 - 2.0 ms as **baseline**.

We will use a sliding window approach to approximate the **gradient** of the linear slope within the time 0.5 - 1.2 ms.

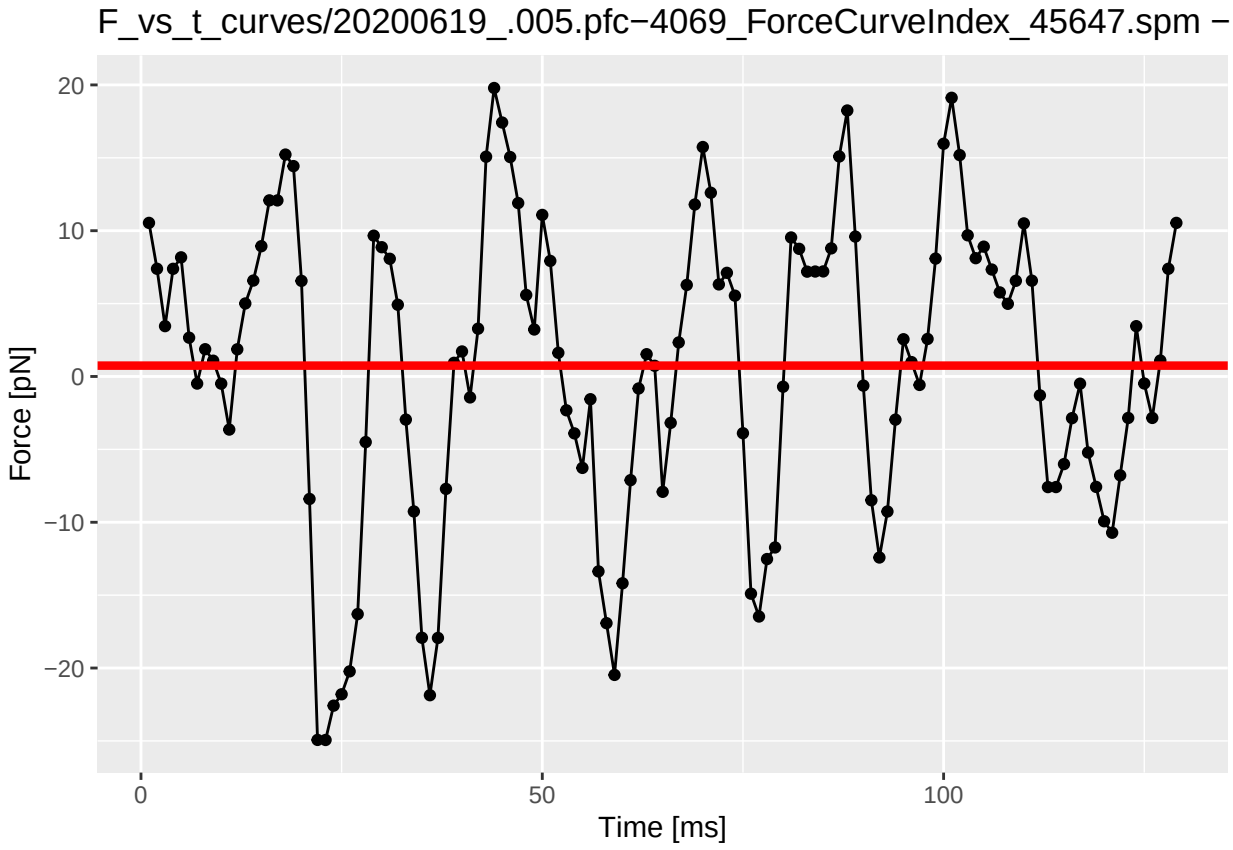
Extract maximal force (F max) from each graph

F max = 149.9152, the time of F max = 0.9843756.

Compute Contacting point

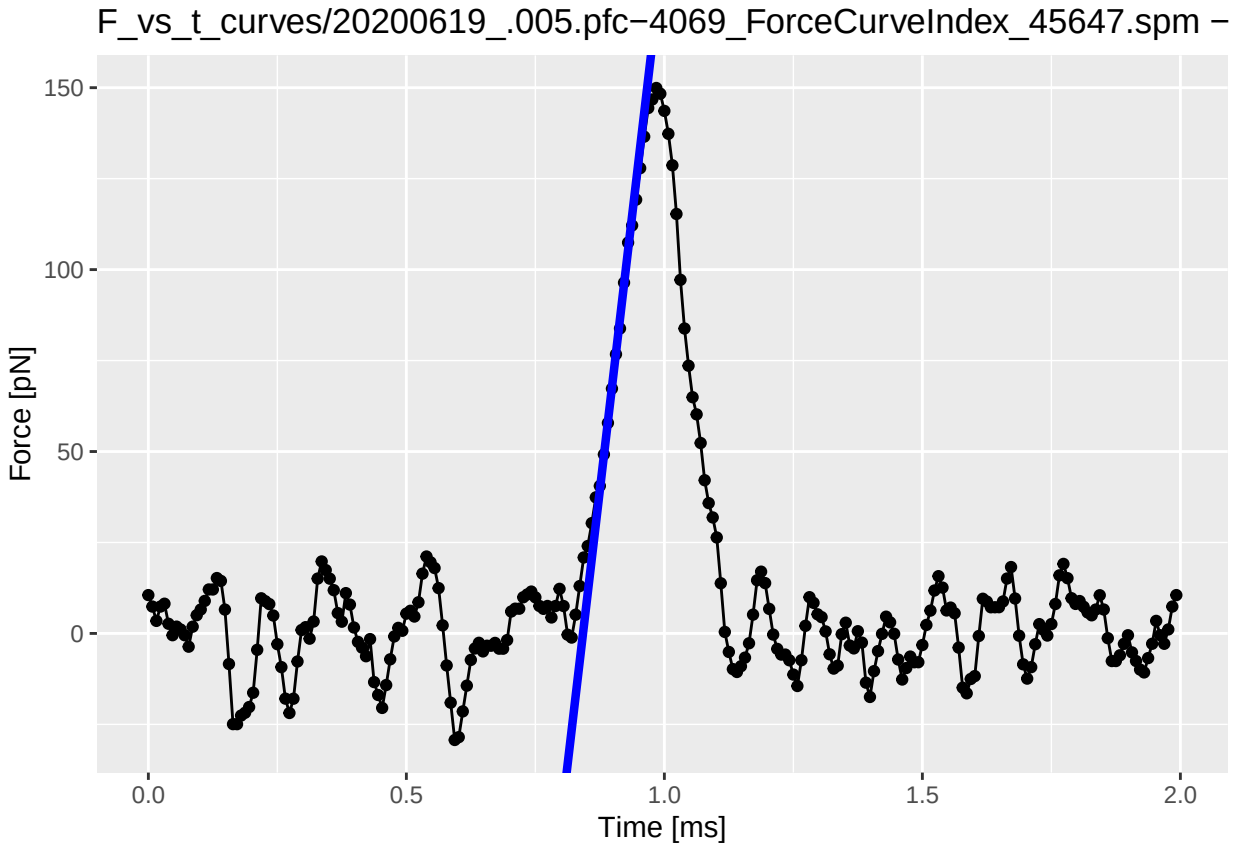
A) Intersect between baseline and linear gradient

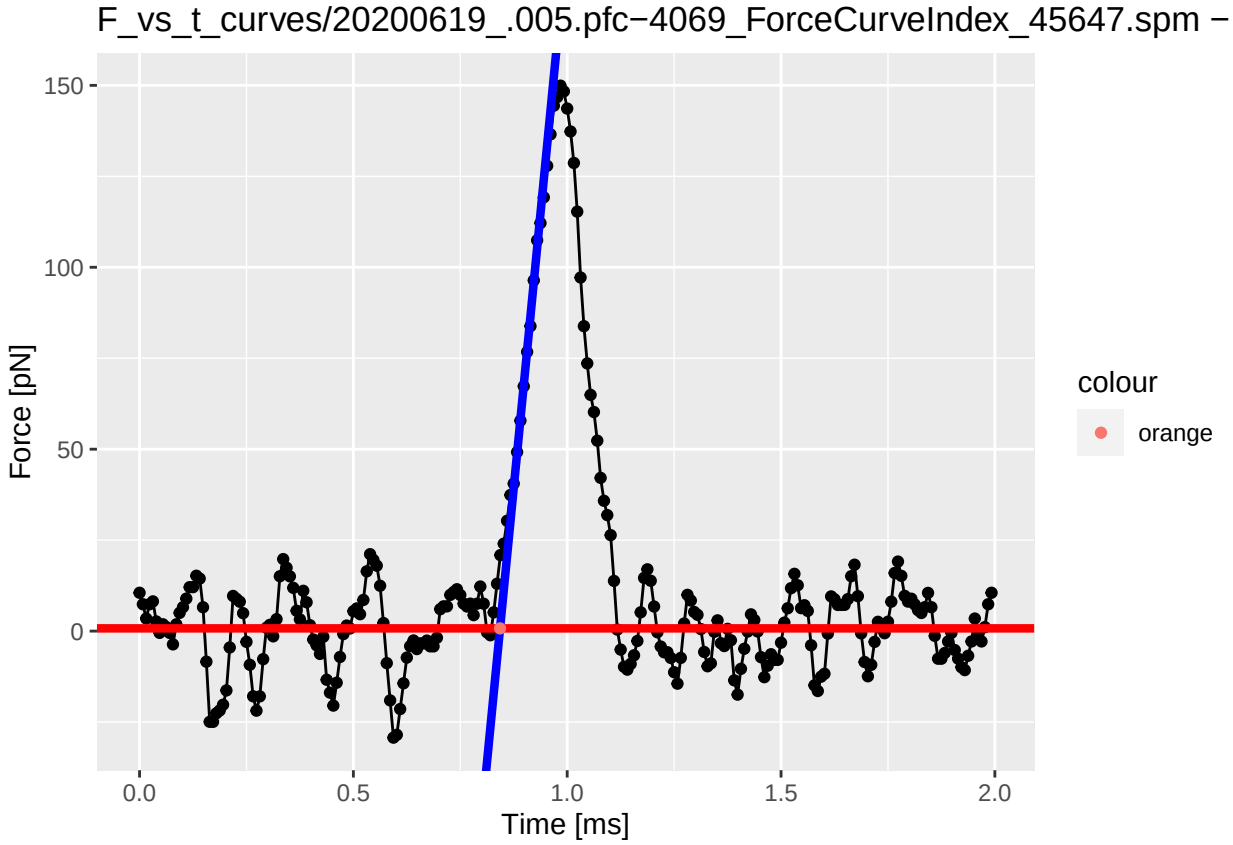
- We will use the force signal between time 0.0 - 0.5 ms as well as 1.5 - 2.0 ms as **baseline**.
- We will use a sliding window approach to approximate the **gradient** of the linear slope within the time 0.5 - 1.2 ms.




```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

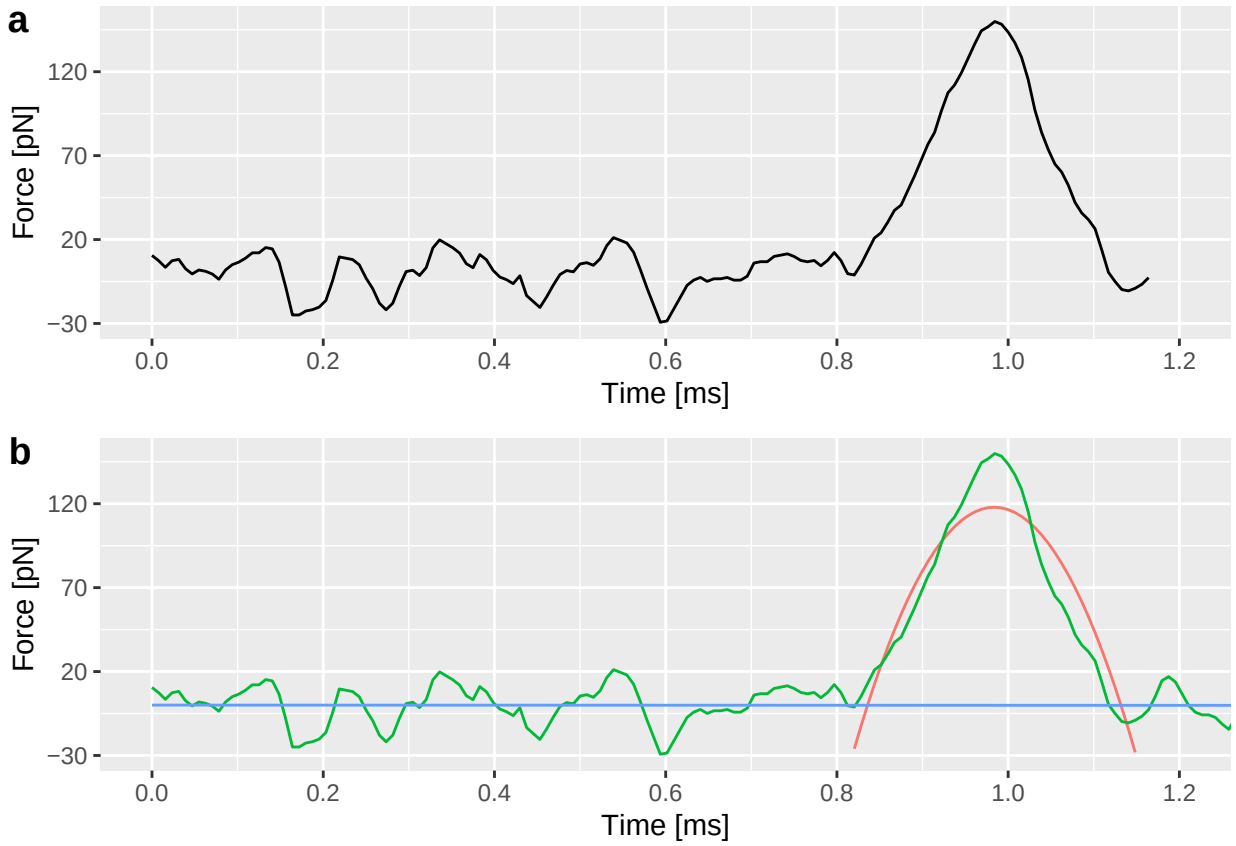

F_vs_t_curves/20200619_.005.pfc−4069_ForceCurveIndex_45647.spm −
150
100
50
0
Force [pN]
0.0
0.5
1.0
1.5
2.0
Time [ms]


F_vs_t_curves/20200619_.005.pfc−4069_ForceCurveIndex_45647.spm −
150
100
50
0
Force [pN]
0.0
0.5
1.0
1.5
2.0
Time [ms]
colour
orange

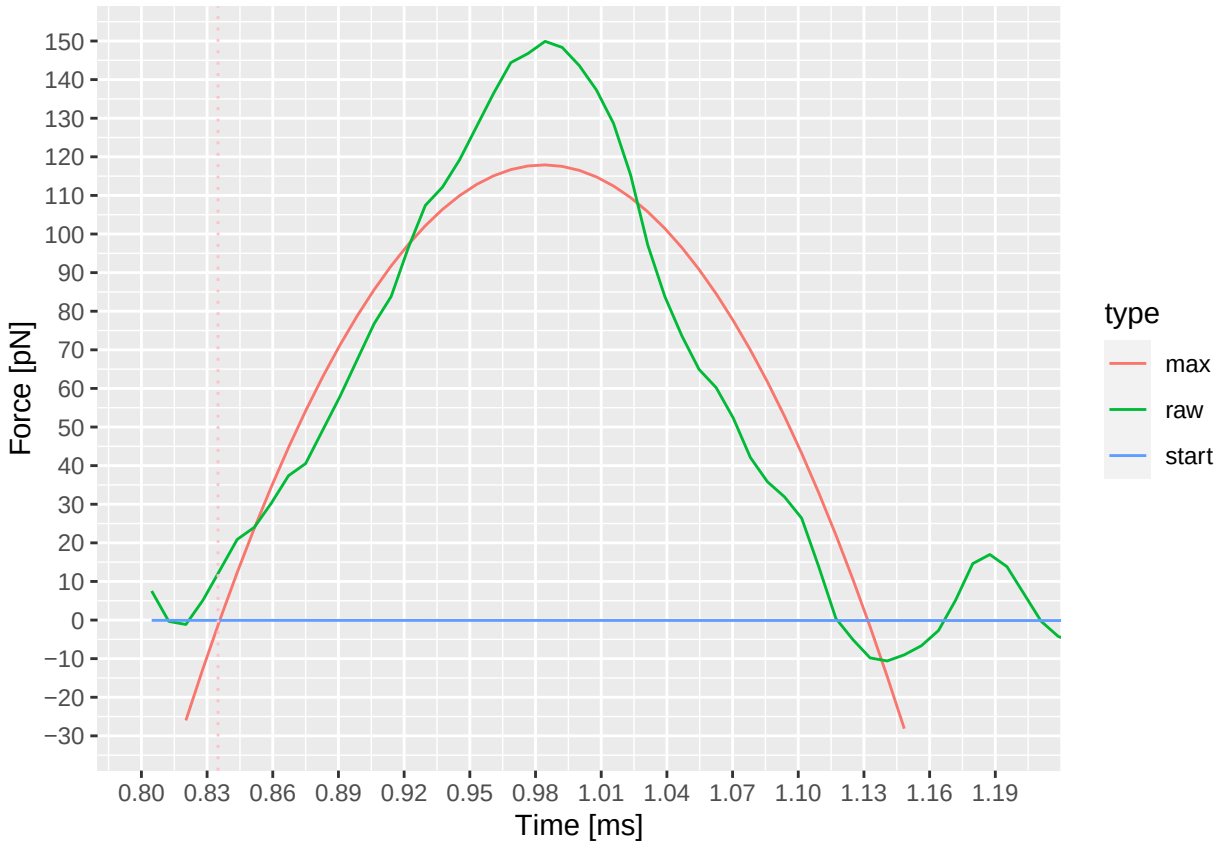
B) Intersect between baseline and parabular fit & comparison to linear fit intersect

```
# Audrey

### create long form ###
#min(df$pN[105])
# dfmax = df[110:150,]
# dfstart = df[0:100,]
# modelmax = lm(pN ~ poly(ms,2), data= dfmax)
# modelstart = lm(pN ~ ms, data= dfstart)
# ms = seq(0, 2, 0.01)
# df$predicted.intervals_max <- predict(modelmax, df)
# df$predicted.intervals_start <- predict(modelstart, df)
# dflong = data.frame(ms = df$ms,
#             pN = data.matrix(c(df$pN, df$predicted.intervals_start,
#                                df$predicted.intervals_max)),
#             type = as.factor(c(rep("raw", length(df$pN)),
#                                rep("start", length(df$predicted.intervals_start)),
#                                rep("max", length(df$predicted.intervals_max)))))
#save(dflong, file = "F_vs_t_curves/dflong.RData")
#########################

df = F_vs_t_curve1
p1 <- ggplot(df[1:150,]) +
  geom_line(aes(x = ms, y = pN)) +
  labs(x = "Time [ms]",
      y = "Force [pN]") +
  coord_cartesian(xlim=c(0.0, 1.2), ylim = c(-30,150)) +
  scale_x_continuous(limits = c(0,2), breaks= seq(0,1.2,0.2)) +
  scale_y_continuous(limits = c(-30, 150), breaks = seq(-30, 150,50)) + theme_gray()

load("F_vs_t_curves/dflong.RData")
p2 <- ggplot(dflong) +
  geom_line(aes(ms, pN, colour = type)) +
  coord_cartesian(xlim=c(0.0, 1.2), ylim = c(-30,150)) +
  scale_x_continuous(limits = c(0,2), breaks= seq(0,1.2,0.2)) +
  scale_y_continuous(limits = c(-30, 150), breaks = seq(-30, 150,50)) +
  labs(x = "Time [ms]",
      y = "Force [pN]") + theme_gray() +
  theme(legend.position = "none")

plot_grid(p1, p2, labels = "auto", ncol = 1)
```

```
## Warning: Removed 213 row(s) containing missing values (geom_path).
```




```
# grid.arrange(arrangeGrob(p1, ncol=1, nrow=1),
#          arrangeGrob(p2, ncol=1, nrow=2), heights = c(1,3))

#intersection is 0.835 ms and 0 pN
```



C) Mean of error increase from baseline (= start of adhesion dent) and error from linear gradient (= end of adhesion dent)

```
# suggested by Jörg Stelling
```

Compute indentation depth (d) from Contacting point for each pixel

The indentation depth d = 3.65451 nm.

Compute modulus (= stiffness, E) for each pixel from F-max and d

The Young's modulus E = 11.9858418 Mega Pascal.

Visualisation of the Young's Modulus

Print picture of topology

Print picture of stiffness

Error propagation/ sensitivity analysis of the modulus

Plot topology against modulus? Can this detect 'antibiotics affected areas'?