

# Particle Interception as a Function of Reynolds Number and Collector Density

Version 3.0

Jordan Wingenroth

8/28/2020

## Contents

Data Tidying . . . . .	1
Modelling for ECE% . . . . .	4
Uncertainty propogation function: mutate_with_error . . . . .	4
k_tot (total time-decay rate) . . . . .	4
k_s (time-decay due to settling) . . . . .	6
k_bg (time-decay due to settling in the rest of the flume) . . . . .	7
k_c (time-decay due to collection) . . . . .	8
eta . . . . .	9
Addressing Laurel's comments on the outline . . . . .	10
1: Finding $C$ for each level of $ad$ , comparing to Fauria . . . . .	10
2: Comparing Fauria model predictions to our observations . . . . .	13
3: Monte Carlo approach . . . . .	14
3 addendum: Safety check on null result (assuming no effect of treatments on $k_c$ ) . . . . .	16

## Objective(s)

In the interest of documenting our analytic methods, I've decided to make one comprehensive, hopefully lay-readable document including all steps from our raw laboratory data to the model results. I won't be able to include the turbulence analyses in time for our meeting (8/27), but that required Laurel's Matlab script anyways.

## Data Tidying

### CHANGES FROM VERSION 1 (8/26)

I pared back this section to reduce the page count. Please refer to the original version to see visualizations and more detailed steps and notes.

### Time-decay runs:

We entered data into Google Sheets as we processed samples in the lab. I downloaded .csv versions of these data in 2019, then made a few modifications as I explored confounders. Today, I added data from runs done in my absence to this dataset. Field names, date syntax, and other conventions weren't kept consistent, so some data wrangling is necessary. I also removed the blank data at the beginning of some files manually.

```
library(tidyverse)
library(broom)
library(cowplot)
library(lme4)
```

```
pumpfiles <- list.files("../data/peristaltic pumps/")
trapfiles <- list.files("../data/sediment traps/")
```

```
pumpdate <- str_sub(pumpfiles,0,6)
trapdate <- str_sub(trapfiles,0,6)
```

In all, five runs are missing sediment trap data, meaning they won't be used in our final, published analyses. Let's turn our attention to the suspended concentration data (i.e., pump data).

```
pumpdata <- lapply(pumpfiles, function(x) read_csv(paste0("../data/peristaltic pumps/",x)))

names(pumpdata) <- pumpdate

x <- pumpdata

tidypump <- lapply(seq_along(x), function(i) {
  select(x[[i]],
    loc = Location,
    ht = Height,
    t = `time series`,
    mvc = contains("(ppm)")) %>%
  filter(t < 21) %>% #filter a few timepoints outside the normal window
  mutate(t = (t-min(t)+1)*300, #convert from timestep to seconds
    mvc = as.numeric(mvc),
    date = as.numeric(names(x)[[i]])) %>%
  filter(mvc<80, mvc>8) #outliers were removed based on the residual graph
})
)

pump <- bind_rows(tidypump)
```

So now we have a long table of our pump data (time  $\times$  concentration) stratified by run (i.e., treatment), height, and upstream/downstream location.

Aside from a few runs where **part** or **all** of the data don't follow the decay pattern, the results look pretty good! If we repeated runs as necessary to replace those with erroneous data, that shouldn't be a problem. Here I'll narrow down our dataset by knocking out runs with clear issues.

```
pump <- pump %>%

#first 3 runs had starting sediment mass of 100g rather than 200g

  filter(date > 181005) %>%

#we can't use runs without sediment mass

  filter(as.character(date) %in% trapdate)
```

So this leaves 20 rows. Let's join this to our run metadata table.

```
metadata <- read_csv("../data/run_metadata.csv")

pump <- left_join(pump, metadata, by = "date")
```

Now then, let's remove the biofouled runs and a couple runs with other issues, which won't be used in assessing our primary hypothesis about Reynolds number and collector density, and we'll see what we're left

with.

```
pumpfinal <- pump %>%  
  filter(growth_days==0, !date %in% c(190417, 190802, 190926), # leaving out old control and  
         dowl_density != "0232")
```

There are certainly no trends jumping out immediately, which isn't necessarily worrisome since exponential decay is hard to compare by eye. More on that later though.

For the sediment traps, data import and wrangling follows a similar structure to the pump data.

```
trapdata <- lapply(trapfiles, function(x) read_csv(paste0("../data/sediment traps/",x)))  
  
names(trapdata) <- trapdate  
  
x <- trapdata  
  
tidytrap <- lapply(seq_along(x), function(i) {  
  select(x[[i]], station = 1, pre = contains("pre"), post = contains("post"), sed = contains("sed")) %>%  
  mutate(date = names(x)[i]) %>%  
  mutate_at(vars(pre,post,sed,date), as.numeric)  
})  
  
trap <- bind_rows(tidytrap) %>%  
  filter(!is.na(sed))
```

So, across 21 runs, we had 186 samples. We were set up to collect 9 per run, but traps broke, filters slipped and spilled sediment, et cetera.  $186/21 = 8.8571429$ , so our success rate was actually pretty good, although some of those measurements were probably erroneous despite being measured.

```
trapfinal <- left_join(trap, metadata, by = "date") %>%  
  filter(date %in% pumpfinal$date)
```

### Velocity experiment:

We estimated Reynolds number from a flow velocity experiment in the open channel conducted on Jan 28, 2019. The Vectrino data is in commit [e78357](#) and the R code for the regression in commit [17c8af](#) in the [esdlflume](#) Github repo (Thanks Candace!).

There's an image in version 1 (and in `/pics`) but the linear regression is simply  $u = 0.00081282 + 0.00195723f$ , where  $u$  is flow velocity (m/s) and  $f$  is pump frequency (Hz). Hence our velocities at 10, 20, and 30 Hz, were 0.0203851, 0.0399574, and 0.0595297 respectively.

It should be noted that the back-of-envelope estimate of  $v = f/500$  is very close for our range, probably by design. For the regression,  $R^2 > .99$ .

### Flume volume experiment:

The test section is modeled as a rectangular prism, making calculation of volume trivial. However, the water in the flume is in an irregular form. In order to correct for the time water spends flowing outside the test section, where collectors are not acting on particles, we must estimate the total volume of water. We did so using a simple integration of the volumetric flow rate at the drain. We ran an experiment previously in 2018 or 2019, but my physics common sense lapsed: We picked the drain hose up off the ground to take measurements, then stuck it back down in the drain for most of the flow time. This led to an underestimate of about 5-10%. I re-did the analysis. 1st through 4th degree polynomial regression all yield the same volume +/- <0.5% (code in this repo, under `/code`)

I rounded off to  $2.43 \text{ m}^3$ . The test section has been measured as  $1.95 \times .6 \times .4 = 0.468 \text{ m}^3$ . Hence the corrective factor is 5.1923077.

## Modelling for ECE%

So at this point, we really only need `pumpfinal`, `trapfinal`, maybe `metadata` for good measure, the frequency-to-velocity regression coefficients, and individual constants about the physical setup such as dowel diameter, flume volume, starting sediment mass, dynamic viscosity etc.

I'll also mention here that our model is based on the general equation:

$$k_{tot} = k_s + k_c + k_{bg}$$

where the  $k$ 's represent total decay rate and portions of it due to settling, collection, and (background) settling in the rest of the flume outside the test section, respectively.

### Uncertainty propogation function: `mutate_with_error`

Here is a function found on StackExchange.com, which performs error propogation in R using derivatives. Error values should be named "dx", with x being the corresponding variable name.

```
mutate_with_error = function(.data, f) {
  exprs = list(
    # expression to compute new variable values
    deparse(f[[3]]),

    # expression to compute new variable errors
    sapply(all.vars(f[[3]]), function(v) {
      dfdp = deparse(D(f[[3]], v))
      sprintf('(d%s*(%s))^2', v, dfdp)
    }) %>%
    paste(collapse='+') %>%
    sprintf('sqrt(%s)', .)
  )
  names(exprs) = c(
    deparse(f[[2]]),
    sprintf('d%s', deparse(f[[2]]))
  )

  .data %>%
    # the standard evaluation alternative of mutate()
    mutate_(.dots=exprs)
}
```

### `k_tot` (total time-decay rate)

We estimate `k_tot` from our pump data by linear regression of `log(mvc)`:

```
library(broom)

r2mod <- list()
r2val <- list()
ktval <- list()

for (k in 1:length(unique(pumpfinal$date))) {
  tempdata <- pumpfinal %>%
    filter(date == unique(pumpfinal$date)[k])

  r2mod[[k]] <- lm(data = tempdata, formula = log(mvc) ~ t)
  r2val[[k]] <- summary(r2mod[[k]])$r.squared
}
```

```

ktval[[k]] <- -coef(r2mod[[k]])[2]
}

names(r2mod) <- unique(pumpfinal$date)
names(r2val) <- unique(pumpfinal$date)
names(ktval) <- unique(pumpfinal$date)

r2final <- tibble(unlist(r2val), names(unlist(r2val)), unlist(ktval)) %>% #requires single digit start
  transmute(r2 = .[[1]],
            kt = .[[3]],
            start = str_sub(.[[2]], 6,6),
            stop = str_sub(.[[2]], 12,12),
            date = as.numeric(str_sub(.[[2]], 14,))) %>%
  left_join(metadata, by = "date")

ktdata <- bind_rows(lapply(r2mod, tidy), .id = "date") %>%
  filter(term == "t")

```

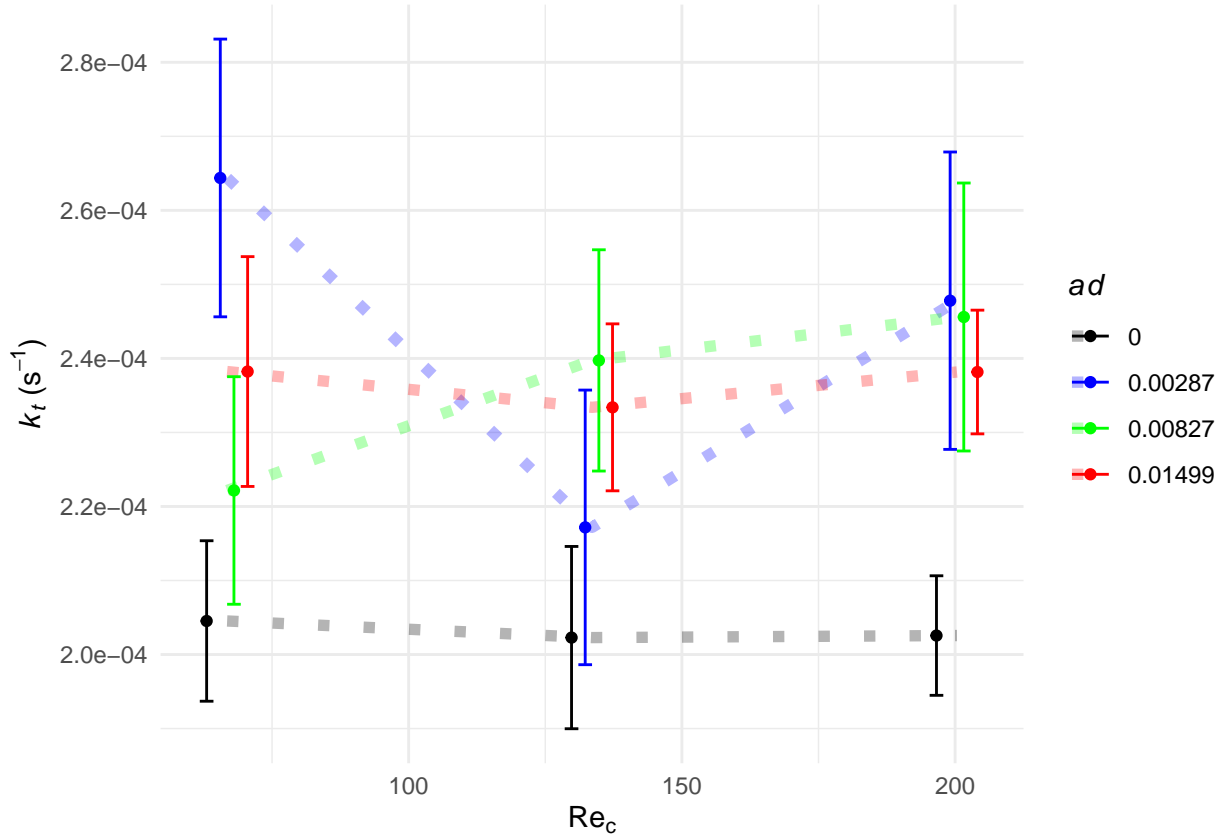
So then, `k_tot` as a function of our independent vars:

```

fitdata <- ktdata %>%
  mutate(k_t = -estimate, dk_t = std.error, date = as.numeric(date)) %>%
  left_join(metadata, by = "date") %>%
  mutate(ad = as.numeric(dowel_density)*.003175^2*2/1.95, Re = pump_freq/500*.003175/9.509e-7) # DOWEL

fitdata %>%
  ggplot(aes(x = Re, y = k_t, ymin = k_t - 1.96*dk_t, ymax = k_t + 1.96*dk_t, color = as.character(round(
  geom_line(lty = "dotted", size = 2, alpha = .3) +
  geom_point(position = position_dodge(width = 10)) +
  geom_errorbar(width = 10, position = "dodge") +
  scale_color_manual(values = c("black","blue","green","red")) +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE)) +
  labs(y = expression(paste(italic(k[t]), " (" , s^{-1} , ")")), x = expression(Re[c]), color = expression(i
  theme_minimal()

```



$k_s$  (time-decay due to settling)

$$m_s = \frac{k_s}{k_{tot}}(1 - e^{-kT})m_0$$

Or, expressed for  $k_s$ ,

$$k_s = \frac{m_s}{m_0} \left( \frac{1}{1 - e^{-kT}} \right) k_{tot}$$

where  $m_s$  is mass settled in the test section (estimated from sediment traps),  $m_0$  is starting sediment mass (200 g for this study), and  $T$  is the total exposure time of the sediment traps (100 min, i.e., 6000 s, for this study).

```
final <- trapfinal %>%
  group_by(date) %>%
  summarise(m_trap = mean(sed)/1000,
            dm_trap = sd(sed)/1000) %>% #average sediment in trap converted from mg to g
  mutate_with_error(m_s ~ m_trap*1.95*.6/(3.14159*.0127^2)) %>% ##A_TS/a_trap
  left_join(fitdata, by = "date") %>%
  mutate_with_error(c_s ~ m_s/200/(1-exp(-k_t*6000))) %>% #intermediate step to squash error
  mutate_with_error(k_s ~ c_s*k_t)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## Warning: `mutate_()` is deprecated as of dplyr 0.7.0.
```

```
## Please use `mutate()` instead.
```

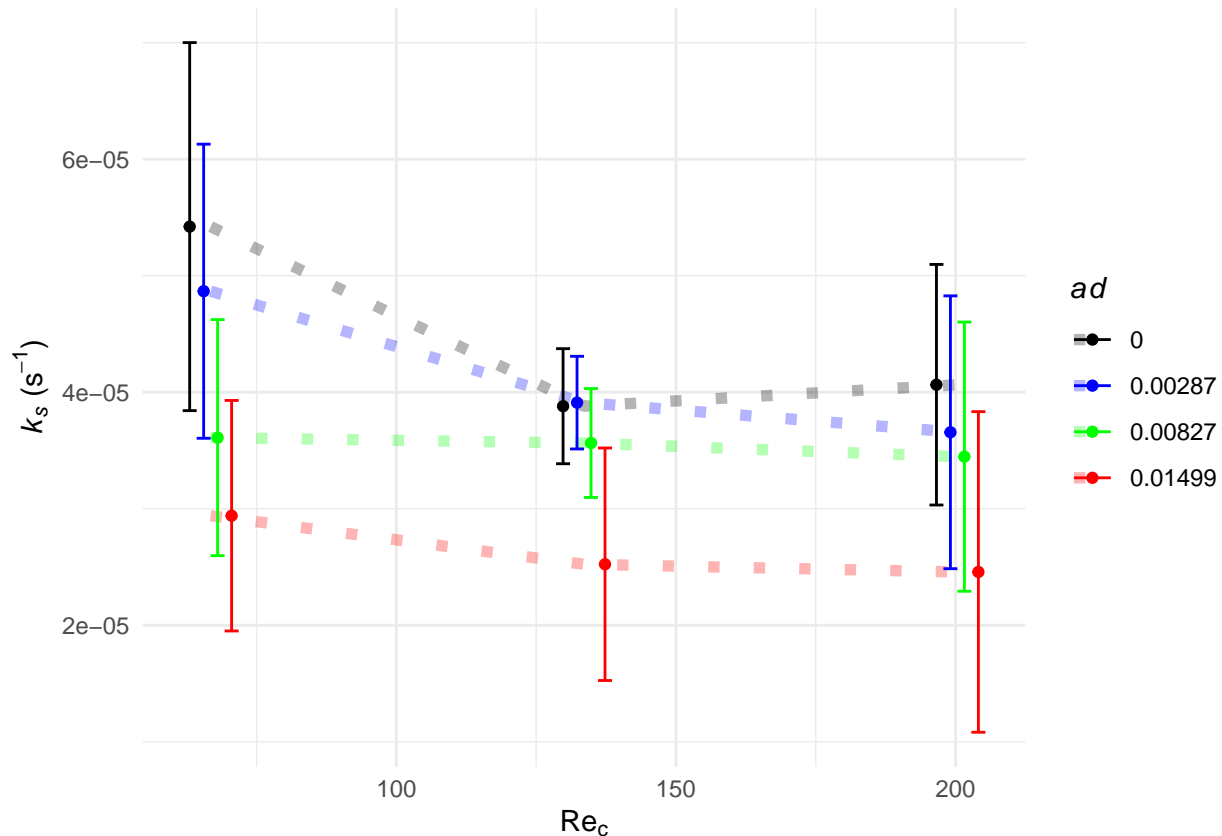
```
## See vignette('programming') for more help
```

```
## This warning is displayed once every 8 hours.
```

```
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
final %>%
```

```
ggplot(aes(x = Re, y = k_s, ymin = k_s - dk_s, ymax = k_s + dk_s, color = as.character(round(ad,5)))) +
  geom_line(lty = "dotted", size = 2, alpha = .3) +
  geom_point(position = position_dodge(width = 10)) +
  geom_errorbar(width = 10, position = "dodge") +
  scale_color_manual(values = c("black", "blue", "green", "red")) +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE)) +
  labs(y = expression(paste(italic(k[s]), " (" , s^{-1} , ")")), x = expression(Re[c]), color = expression(i
  theme_minimal()
```



$k_s$  seems negatively related to both dowel density and flow velocity.

**k<sub>bg</sub> (time-decay due to settling in the rest of the flume)**

In control runs,  $k_c = 0$ , so we can estimate the decay rate in the rest of the flume as  $k_{bg} = k_{tot} - k_s$

```
bgvals <- final %>%
```

```
  filter(dowel_density=="0000") %>%
  mutate_with_error(k_bg ~ k_t - k_s) %>%
  select(pump_freq, k_bg, dk_bg) %>%
  arrange(pump_freq)
```

```
bgvals
```

```
## # A tibble: 3 x 3
```

```
##   pump_freq    k_bg    dk_bg
```

```
##      <dbl>      <dbl>      <dbl>
## 1      10 0.000150 0.0000167
## 2      20 0.000163 0.00000799
## 3      30 0.000162 0.0000111
```

To add it to the table:

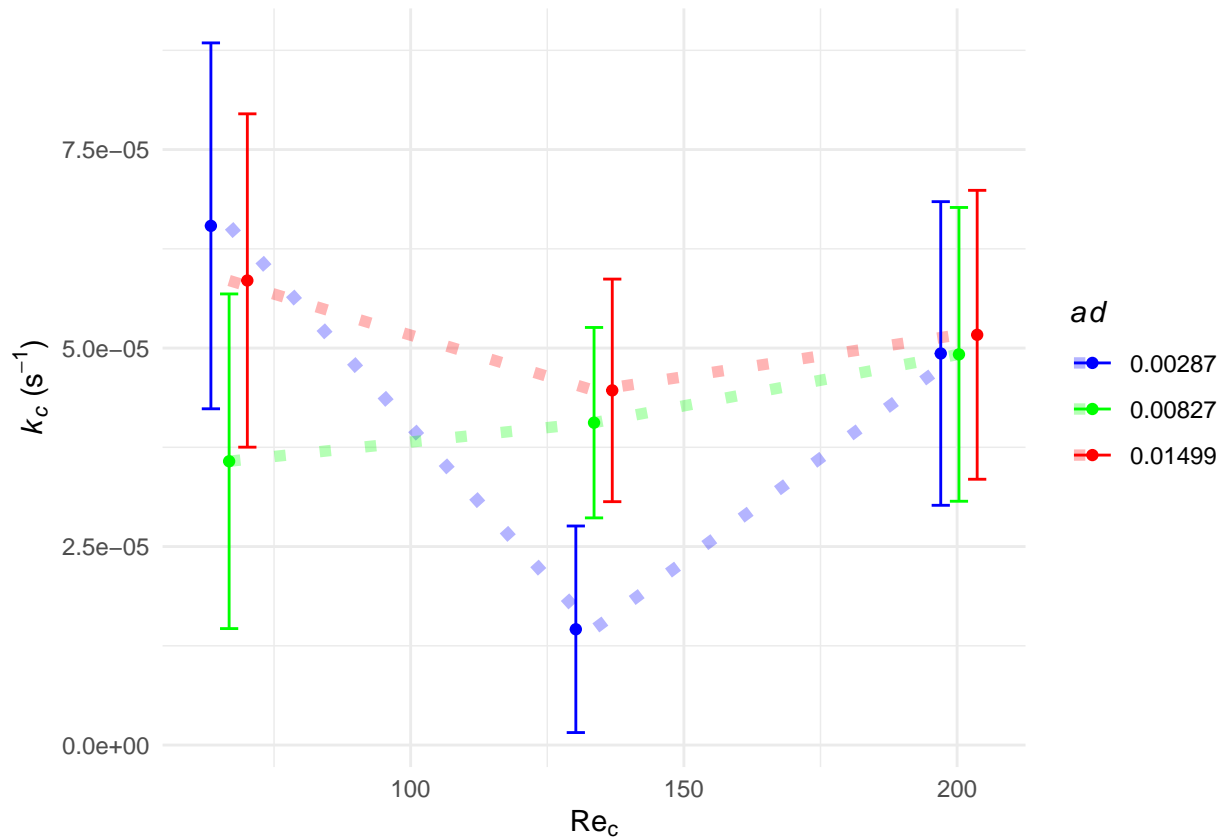
```
final <- left_join(final, bgvals, by = "pump_freq")
```

### k<sub>c</sub> (time-decay due to collection)

So, now that we have our values in the table, k<sub>c</sub> is just a subtraction away:

```
final <- final %>%
  mutate_with_error(k_c ~ k_t - k_s - k_bg)

final %>%
  filter(dowel_density != "0000") %>%
  ggplot(aes(x = Re, y = k_c, ymin = k_c - dk_c, ymax = k_c + dk_c, color = as.character(round(ad,5))))
  geom_line(lty = "dotted", size = 2, alpha = .3) +
  geom_point(position = position_dodge(width = 10)) +
  geom_errorbar(width = 10, position = "dodge") +
  scale_color_manual(values = c("blue","green","red")) +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE)) +
  labs(y = expression(paste(italic(k[c]), " (" , s^{-1} , ")")), x = expression(Re[c]), color = expression(i
  theme_minimal()
```





eta

```
#temp in flume measured at 22.2C with a calibrated thermometer
#plugged into https://www.engineeringtoolbox.com/water-dynamic-kinematic-viscosity-d_596.html
visc = 9.509e-7 #kinematic viscosity, m2/s

d = .003175 #dowel diameter

final <- final %>%
  mutate(frontal_area = as.numeric(dowel_density)*.003175,
         u = as.numeric(pump_freq)/500, #velocity
         Re = u*.003175/visc, #Reynolds #
         k_c, k_s, k_bg) %>%
  mutate(eta = k_c/u/frontal_area,
         deta = dk_c/u/frontal_area) %>%
  mutate(eta = eta * 2.43/(1.95*.4*.6),
         deta = deta * 2.43/(1.95*.4*.6)) # don't forget to correct for time out of test section!

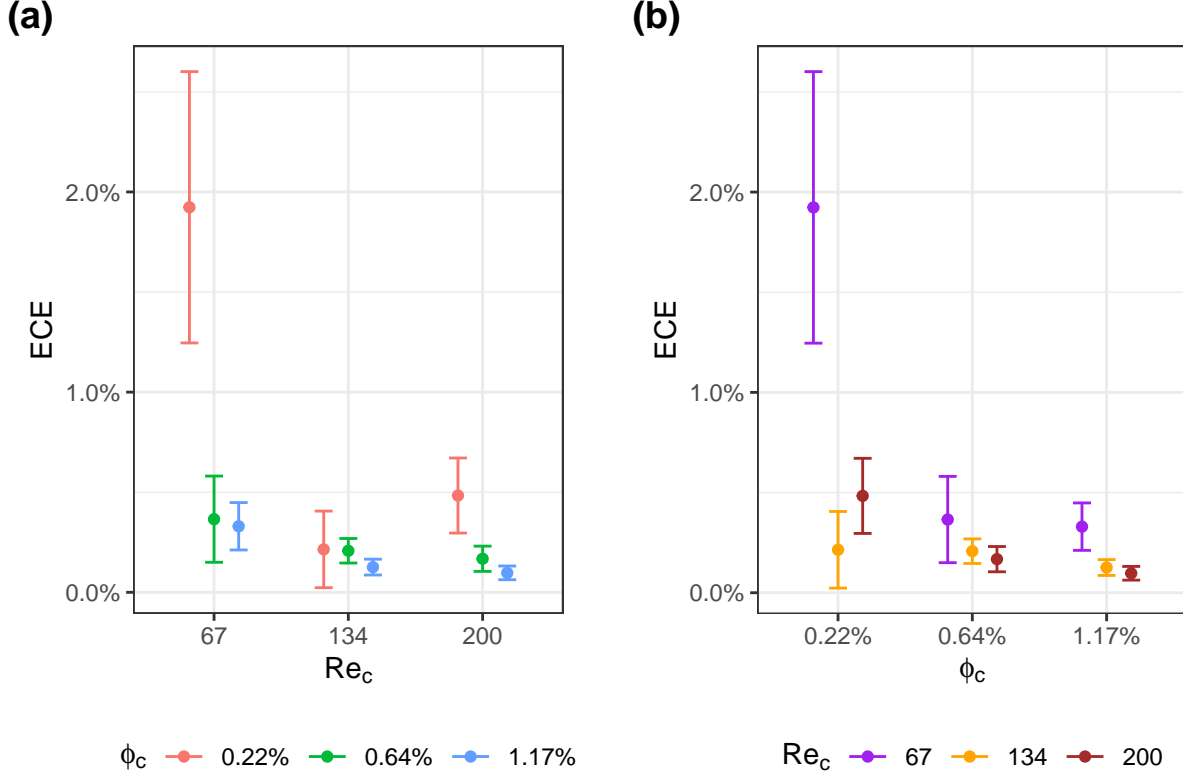
plotece1 <- final %>%
  filter(dowel_density != "0000") %>%
  mutate(Refac = factor(Re, labels = c("Re = 66.78", "Re = 133.56", "Re = 200.34"))) %>%
  ggplot(aes(x = factor(round(Re, 0)), y = eta, ymin = eta - deta, ymax = eta + deta, color = factor(ad
  geom_point(position = position_dodge(width = .55)) +
  geom_errorbar(width = 0.4, position = position_dodge(width = .55)) +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "ECE", x = expression(Re[c]), color = expression(phi[c])) +
  theme_bw() +
  theme(legend.position = "bottom")

plotece2 <- final %>%
  filter(dowel_density != "0000") %>%
  mutate(Refac = factor(Re, labels = c("Re = 66.78", "Re = 133.56", "Re = 200.34"))) %>%
  ggplot(aes(color = factor(round(Re, 0)), y = eta, ymin = eta - deta, ymax = eta + deta, x = factor(ad
  geom_point(position = position_dodge(width = .55)) +
  geom_errorbar(width = 0.4, position = position_dodge(width = .55)) +
  scale_y_continuous(labels = scales::percent) +
  scale_color_manual(values = c("purple","orange","brown")) +
  labs(y = "ECE", color = expression(Re[c]), x = expression(phi[c])) +
  theme_bw() +
  theme(legend.position = "bottom")

plotece <- plot_grid(plotece1, plotece2, labels = c("(a)","(b)"), scale = .9, align = "lrb")

save_plot("../pics/ece_plot.png",plotece, dpi = 600)

plotece
```



## Addressing Laurel's comments on the outline

Most of Laurel's comments addressed the structure and written content of the paper. However, several recommended new analyses to include.

### 1: Finding $C$ for each level of $ad$ , comparing to Fauria

#### COMMENT:

To better flesh out the role of stem density, you could consider figuring out what  $C$  would be for each stem density, using the same functional form of the equation as Fauria.  $C$  from our study and Fauria's study could be plotted against dimensionless stem density (i.e.,  $ad$  or solid volume fraction) to determine the nature of the relationship.

#### ADDRESSED:

Fauria's functional form was  $CRe_c^{-1.14}R^{0.65}$ . Excluding runs with biofilm, they got  $C$  values of 1.53 and 3.18 for stem densities of 7209 (high) and 2724 (low) per meter, respectively.

Let's dig into stem density, frontal area,  $ad$ , etc.

Fauria didn't use cylinders but they give dimensions for their synthetic grass: average x-y plane cross-section was 3mm x 1mm, and it seems that they set water height to match stem height roughly. Given that their stems faced randomly in all directions, average *frontal* diameter per stem assuming a 3mm x 1mm rectangular prism (to avoid a multiple integral) would be approximated by, in mm,

$$\frac{\int_0^{\pi/2} [3\sin(\theta) + \cos(\theta)] d\theta}{\pi/2} = \frac{(1) - (-3)}{\pi/2} = \frac{8}{\pi}.$$

This, converted to meters, times stems per unit bed area gives frontal area per unit volume:  $6.93 \text{ m}^{-1}$  (low) and  $18.36 \text{ m}^{-1}$  (high).

$ad$  is frontal area per unit volume times diameter, making it dimensionless. Since it's unclear which  $d_c$  to use for non-cylindrical objects, I'll just use that same frontal  $d_c$  again:  $8/\pi$ . This yields  $ad$  values of 0.0177 and 0.0467.

Solid volume fraction ( $\phi$ ) is actually quite easy for rectangular prisms, and would theoretically be equivalent for the stems' actual, more trapezoidal shapes, if Fauria averaged correctly. It's simply  $l \times w \times I_c$ . This yields  $\phi$  values of 0.00817 and 0.02163.

As for our values, cylinders make everything quite easy. We already have stem density and  $ad$  in the table, and can calculate frontal area and  $\phi$  with ease:

```
final <- final %>%
  mutate(frontal = ad/.003175, phi = ad*pi/4)

final %>%
  select(dowel_density, frontal, ad, phi) %>%
  distinct() %>%
  arrange(ad) %>%
  filter(ad>0)
```

```
## # A tibble: 3 x 4
##   dowel_density frontal      ad      phi
##   <chr>          <dbl>   <dbl>   <dbl>
## 1 0278          0.905 0.00287 0.00226
## 2 0800          2.61 0.00827 0.00650
## 3 1450          4.72 0.0150 0.0118
```

Next, particle ratios: Fauria ran models across a range, but without LISST data, we only know  $\eta$  for our particles altogether. Given our constant  $d_c$  of 0.3175 cm and our median particle size of 25.2 microns, we'd get a particle-to-stem ratio of 0.007937. This raised to the .65 power gives a corrective term of 0.04312975.

OK, so now for the C value models:

```
cmodes <- final %>%
  mutate(coeff = 0.04312975*Re^-1.14) %>%
  lmList(eta ~ 0 + coeff|phi, data = .)
```

```
## Warning: Unknown or uninitialised column: `(weights)`.
## Warning: Unknown or uninitialised column: `(offset)`.
## Warning: Unknown or uninitialised column: `(weights)`.
## Warning: Unknown or uninitialised column: `(offset)`.
## Warning: Unknown or uninitialised column: `(weights)`.
## Warning: Unknown or uninitialised column: `(offset)`.
## Warning: Unknown or uninitialised column: `(weights)`.
## Warning: Unknown or uninitialised column: `(offset)`.

## Warning in lmList(eta ~ 0 + coeff | phi, data = .): 1 error caught in lm.fit(x,
## y, offset = offset, singular.ok = singular.ok, ...): 0 (non-NA) cases
```

So then, to put our results and Fauria's side by side:

```
c_over_phi <- cbind(rownames(coef(cmodes))[2:4], coef(cmodes)[2:4,]) %>%
  as.tibble() %>%
```

```

mutate_all(as.numeric) %>%
mutate(lab = "This study") %>%
rename(phi = V1, C = V2)

## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if ` `.name_repair` is
## Using compatibility ` `.name_repair` .
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

fauria <- tibble(phi = c(.00817,.02163), C = c(3.18,1.53), lab = "Fauria et al. (2015)")

c_over_phi <- rbind(c_over_phi, fauria)

lm(data = c_over_phi, formula = log10(C) ~ log10(phi)) %>%
summary

##
## Call:
## lm(formula = log10(C) ~ log10(phi), data = c_over_phi)
##
## Residuals:
##      1      2      3      4      5
## 0.03311 0.05182 0.32871 -0.34621 -0.06744
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0966      0.8243  -2.544   0.0844 .
## log10(phi)   -1.4107      0.3874  -3.642   0.0357 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2806 on 3 degrees of freedom
## Multiple R-squared:  0.8155, Adjusted R-squared:  0.754
## F-statistic: 13.26 on 1 and 3 DF, p-value: 0.0357

c_phi_curve <- tibble(phi = seq(0.002,.03,by = .0001), C = 10^-2.0966*phi^-1.4107)

cplot <- c_over_phi %>%
  ggplot(aes(x = phi, y = C, color = factor(lab, levels = c("This study", "Fauria et al. (2015)")))) +
  geom_line(data = c_phi_curve, aes(x = phi, y = C), inherit.aes = FALSE) +
  geom_point(size = 2, pch = 4) +
  theme_bw() +
  labs(x = expression(phi[c]), color = "") +
  scale_color_manual(values = c("darkgreen","red")) +
  scale_x_continuous(labels = scales::percent) +
  annotate("polygon", x=c(.012,.012,.004,.004,.007), y= c(23,27,27,23,23), color = "black", fill = "gray",
  annotate("text", .008, 25, parse = TRUE, label = "C == 0.00801*phi[c]^-1.41")

ggsave("../pics/cphiplot.png",cplot,dpi = 600, height = 4, units = "in")

```

```
## Saving 6.5 x 4 in image
```

## 2: Comparing Fauria model predictions to our observations

### COMMENT:

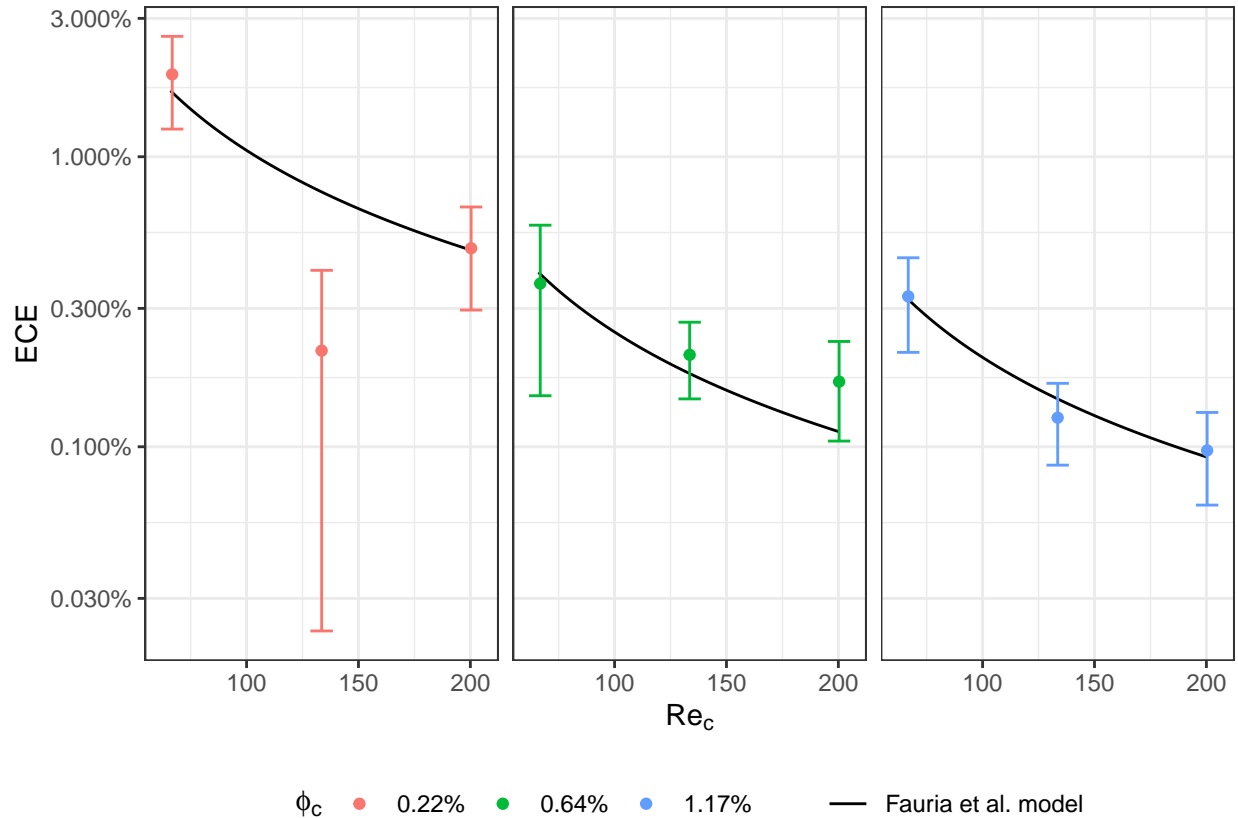
If you use the same exponents as Fauria but treat  $C$  as a free parameter for each stem density configuration, how do predictions compare to the observations? This would be a plot either of  $Re$  vs  $\eta$  for experimental data (points with error bars) and model predictions (lines) or of predicted vs. observed  $\eta$  (with error bars for observed).

### ADDRESSED:

```
# curve data
fauria_pred <- tibble(Re = rep(seq(66,200,length = 1000),3),
  C = c(rep(46.46556,1000),rep(10.987587,1000),rep(8.983344,1000)),
  phi = c(rep(0.002257447,1000),rep(0.006496250,1000),rep(0.011774453,1000))) %>%
  mutate(eta_est = Re-1.14*C*0.04312975, ad = phi)

plotcomp <- final %>%
  filter(dowel_density != "0000") %>%
  mutate(Refac = factor(Re, labels = c("Re = 66.78", "Re = 133.56", "Re = 200.34"))) %>%
  ggplot(aes(x = Re, y = eta, ymin = eta - deta, ymax = eta + deta, color = factor(phi, labels = c("0.2", "0.1", "0.05")))) +
  geom_line(data = fauria_pred, aes(y = eta_est, x = Re, linetype = "Fauria et al. model"), inherit.aes = FALSE) +
  geom_point() +
  geom_errorbar(width = 10) +
  scale_y_log10(labels = scales::percent) +
  facet_wrap(~ as.character(round(phi,5))) +
  labs(y = "ECE", x = expression(Re[c]), color = expression(phi[c])) +
  theme_bw() +
  scale_linetype_discrete(name = "") +
  theme(strip.background = element_blank(),
    strip.text.x = element_blank(),
    legend.position = "bottom") +
  guides(fill = guide_legend(override.aes = list(linetype = 0)),
    color = guide_legend(override.aes = list(linetype = 0)))

plotcomp
```



```
ggsave("../pics/comparisonplot.png", dpi = 600, height = 4)
```

```
## Saving 6.5 x 4 in image
```

### 3: Monte Carlo approach

#### COMMENT:

Each data point has a normal distribution associated with the estimate of  $\eta$  (defined by the estimate value and the associated standard deviation, i.e., the error bars). For each stem density, you can do Monte Carlo sampling in which you draw a different value of  $\eta$  from each of the three  $Re$  by sampling from the underlying distributions. For each realization of the three  $\eta$  estimates, you would then do a log transformation, fit a straight line, and save the slope of that line. The result is a distribution of slopes over 1000 or more realizations. If the 95% confidence interval for those slopes includes zero, we cannot say with statistical certainty that there is a negative trend with  $Re$ . You would do this for each stem density. Then, you would evaluate whether there is a significant trend in  $\eta$  with stem density by holding  $Re$  constant and doing the same thing for each  $Re$ . You could show the results as box plots, where you use some symbol (such as an asterisk) to denote those boxes whose 95% confidence interval does not include zero.

#### ADDRESSED:

Models:

```
library(cowplot)

reps = 30000

skeleton_eta <- final %>%
  filter(frontal_area > 0) %>%
```

```

select(phi, Re, eta, deta) %>%
  arrange(phi, Re)

monte <- list()
for (i in 1:nrow(skeleton_eta)) {
  monte[[i]] <- tibble(phi = skeleton_eta$phi[i],
                      Re = skeleton_eta$Re[i],
                      eta = rnorm(reps, mean = skeleton_eta$eta[i], sd = skeleton_eta$deta[i]))
}
lmsphi <- bind_rows(monte) %>%
  mutate(ID = paste0(phi, 1:reps),
         logRe = log(Re)) %>%
  lmList(log(eta) ~ logRe|ID, data = .) %>%
  coefficients()
lmsphi <- tibble(phi = str_sub(rownames(lmsphi),0,6),lmsphi)
lmsphi <- lmsphi %>%
  group_by(phi) %>%
  summarise(avg = mean(logRe, na.rm = TRUE), sd = sd(logRe, na.rm = TRUE)) %>%
  mutate(ymax = avg + 1.96*sd)
lmsphi

lmsre <- bind_rows(monte) %>%
  mutate(ID = paste0(Re, 1:reps),
         logphi = log(phi)) %>%
  lmList(log(eta) ~ logphi|ID, data = .) %>%
  coefficients()

lmsre <- tibble(Re = str_sub(rownames(lmsre),0,6),lmsre)

lmsre <- lmsre %>%
  group_by(Re) %>%
  summarise(avg = mean(logphi, na.rm = TRUE), sd = sd(logphi, na.rm = TRUE)) %>%
  mutate(ymax = avg + 1.96*sd)

lmsre

```

Plots:

```

p_phi <- lmsphi %>%
  ggplot(aes(x = paste0(100*as.numeric(phi),"%"), y = avg, ymin = avg-1.96*sd, ymax = avg+1.96*sd)) +
  geom_point() +
  geom_errorbar() +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw() +
  geom_text(aes(y = avg + 1.96*sd + 1, label = c("*",NA,"*")), fontface = "bold", size = 5) +
  ylim(-3,1.5) +
  xlab(expression(phi[c])) +
  ylab(expression(beta[log(Re[c])]))

p_re <- lmsre %>%
  ggplot(aes(x = as.factor(round(as.numeric(Re),0)), y = avg, ymin = avg-1.96*sd, ymax = avg+1.96*sd)) +
  geom_point() +
  geom_errorbar() +
  geom_hline(yintercept = 0, lty = 2) +

```

```

theme_bw() +
geom_text(aes(y = avg + 1.96*sd + 1, label = c(NA,"*", "*")), fontface = "bold", size = 5) +
ylim(-3,1.5) +
xlab(expression(Re[c])) +
ylab(expression(beta[log(phi[c])]))

monteplot <- plot_grid(p_re, p_phi, labels = c("(a)","(b)"), scale = .9)

monteplot

save_plot("../pics/montecarlo.png", monteplot, dpi = 600)

```

### 3 addendum: Safety check on null result (assuming no effect of treatments on $k_c$ )

#### ALL DATA IN THIS SECTION IS IMAGINARY!!!!

Let's see what results we'd get with these same analyses if we detected no effects of Re or  $\phi$  on  $k_c$ .

```

imaginary_eta <- final %>%
  filter(frontal_area > 0) %>%
  mutate(imagk_c = mean(k_c),
         imagdk_c = mean(dk_c),
         imageta = imagk_c/u/frontal_area,
         imagdeta = imagdk_c/u/frontal_area) %>%
  mutate(imageta = imageta * 2.43/(1.95*.4*.6),
         imagdeta = imagdeta * 2.43/(1.95*.4*.6)) %>%
  select(phi, Re, eta = imageta, imagdeta)

```

Grouped by stem density:

```

monte <- list()

for (i in 1:nrow(imaginary_eta)) {
  monte[[i]] <- tibble(phi = imaginary_eta$phi[i],
                     Re = imaginary_eta$Re[i],
                     eta = rnorm(1000, mean = imaginary_eta$eta[i], sd = imaginary_eta$imagdeta[i]))
}

lmsphi <- bind_rows(monte) %>%
  mutate(ID = paste0(phi, 1:1000)) %>%
  lmList(log(eta) ~ Re|ID, data = .) %>%
  coefficients()

lmsphi <- tibble(phi = str_sub(rownames(lmsphi),0,6),lmsphi)

lmsphi <- lmsphi %>%
  group_by(phi) %>%
  summarise(avg = mean(Re, na.rm = TRUE), sd = sd(Re, na.rm = TRUE)) %>%
  mutate(ymax = avg + 1.96*sd)

lmsphi

lmsphi %>%
  ggplot(aes(x = phi, y = avg, ymin = avg-1.96*sd, ymax = avg+1.96*sd)) +

```



```
geom_point() +  
geom_errorbar() +  
theme_bw()
```