Tricking filter model demo in dynamic Markdown report

Sasha D. Hafner

16 May, 2023

Overview

I am actually using R to build this pdf report from a Markdown-formatted text file, because the task seems so complicated in Python (actually I am not even sure it is possible). The basic idea is simple: combine some descriptive text, Markdown headers and formatting, and Python code in a single text file (here demo.Rmd, for R Markdown). Then run the Python code and combine the output with the Rmd file contents in a pdf or html file. Because this solution uses R, this Rmd file must be processed in R, which will call Python. Collaboration could be challenging with this approach because I had to set the location of the correct Python version below. Presumably this could be set globally somewhere else. Also, the use of R instead of Python for processing complicates things. Anyway, this is one option. . .

Prep

Some R stuff.

```
library(reticulate)
use_python('/usr/bin/python3')
```

And now the Python model

In the Rmd file you have to indicate that a code chunk should be evaluated with Python using

````
#```{python}
#code here. . .
#```
````

Below, I've put most of the Python code in one chunk.

```
# Import necessary packages ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import shutil
import numpy as np
import matplotlib.pyplot as plt

# Import model ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
shutil.copy('../../mod_funcs.py', '.')
```

```
## './mod_funcs.py'
```

```
from mod_funcs import tfmod


# Set model inputs ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# See notes in tfmod.py for more complete descriptions
```

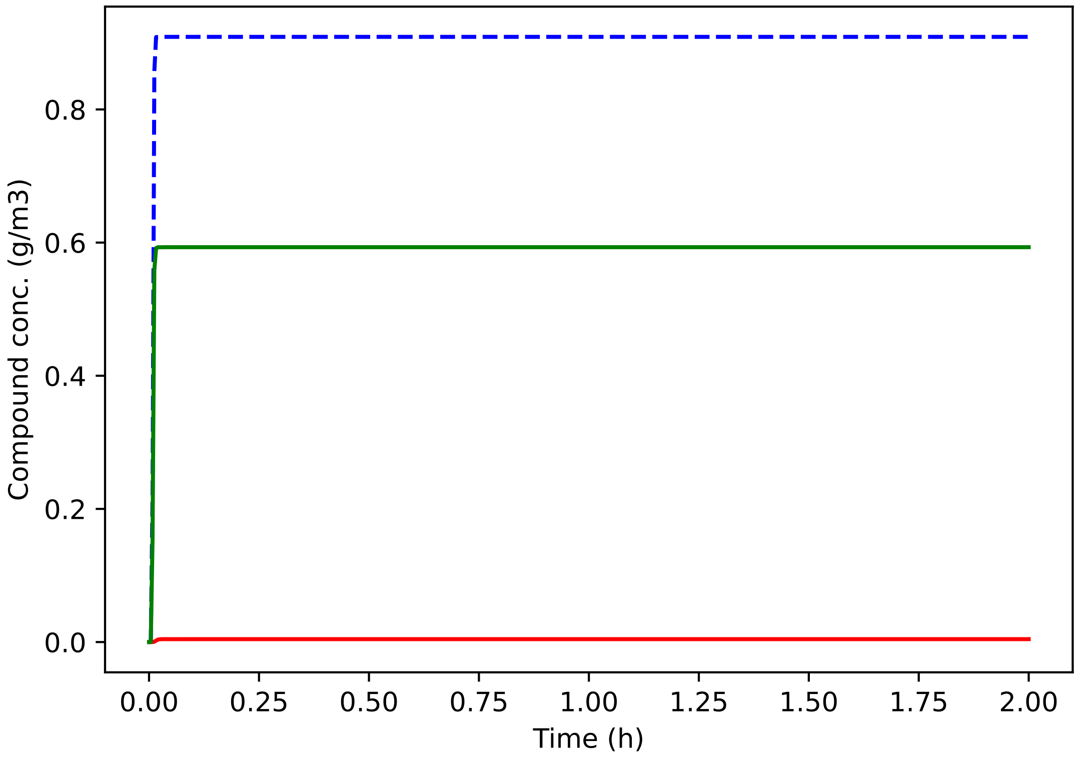
```
L = 2             # Filter length/depth (m)
por_g = 0.5       # (m3/m3)
por_l = 0.25      # (m3/m3)
v_g = 0.03        # Air flow (m/s)
v_l = 2E-5        # Water flow (m/s)
nc = 30           # Number of model cells (layers)
cg0 = 0           # (g/m3)
cl0 = 0           # (g/m3)
cgin = 1.         # Dirty air compound concentration (g/m3)
clin = 0.         # Fresh water concentration (g/m3)
henry = (0.1, 2000.)
temp = 15.        # (degrees C)
dens_l = 1000       # Liquid dens_lity (kg/m3)

k = 500. / 3600

# Low pH to move toward more mass transfer limitation
pH = 4.
pKa = 7.


# Times for model output, calculated from tt and nt here but could be set directly
# Total duration (hours)
tt = 2
# Number of time rows
nt = 500
times = np.linspace(0, tt, nt) * 3600
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Scenarios ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Sim 1 set Kga manually
# Red line
Kga = 0.6
pred1 = tfmod(L = L, por_g = por_g, por_l = por_l, v_g = v_g, v_l = v_l, nc = nc, cg0 = cg0,
              cl0 = cl0, cgin = cgin, clin = clin, Kga = Kga, k = k, henry = henry, pKa = pKa,
              pH = pH, temp = temp, dens_l = dens_l, times = times)

# Sim 2 Use Onda function
# Dashed blue line
pred2 = tfmod(L = L, por_g = por_g, por_l = por_l, v_g = v_g, v_l = v_l, nc = nc, cg0 = cg0,
              cl0 = cl0, cgin = cgin, clin = clin, Kga = 'onda', k = k, henry = henry, pKa = pKa,
              pH = pH, temp = temp, dens_l = dens_l, times = times)

# Sim 3 Use Onda with much higher ssa
# Green line
pred3 = tfmod(L = L, por_g = por_g, por_l = por_l,v_g = v_g, v_l = v_l, nc = nc, cg0 = cg0,
              cl0 = cl0, cgin = cgin, clin = clin, Kga = 'onda', k = k, henry = henry, pKa = pKa,
              pH = pH, temp = temp, dens_l = dens_l, times = times, ssa = 1E6)
```

Plots

Plot outlet concentration (= 1 - removal efficiency here because cgin = 1)

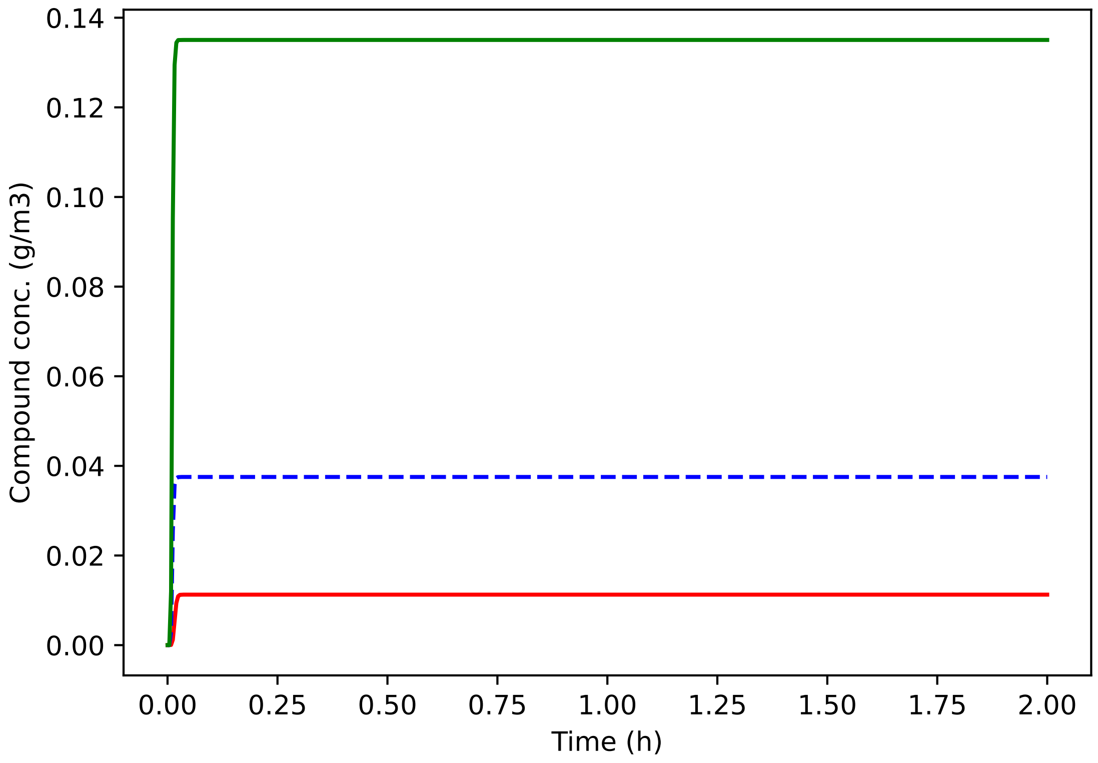
Gas concentration (outlet air)

```
plt.plot(pred1[5] / 3600, pred1[0][nc - 1, :], 'r-')
plt.plot(pred2[5] / 3600, pred2[0][nc - 1, :], 'b', linestyle = 'dashed')
plt.plot(pred3[5] / 3600, pred3[0][nc - 1, :], 'g-')
plt.xlabel('Time (h)')
plt.ylabel('Compound conc. (g/m3)')
```



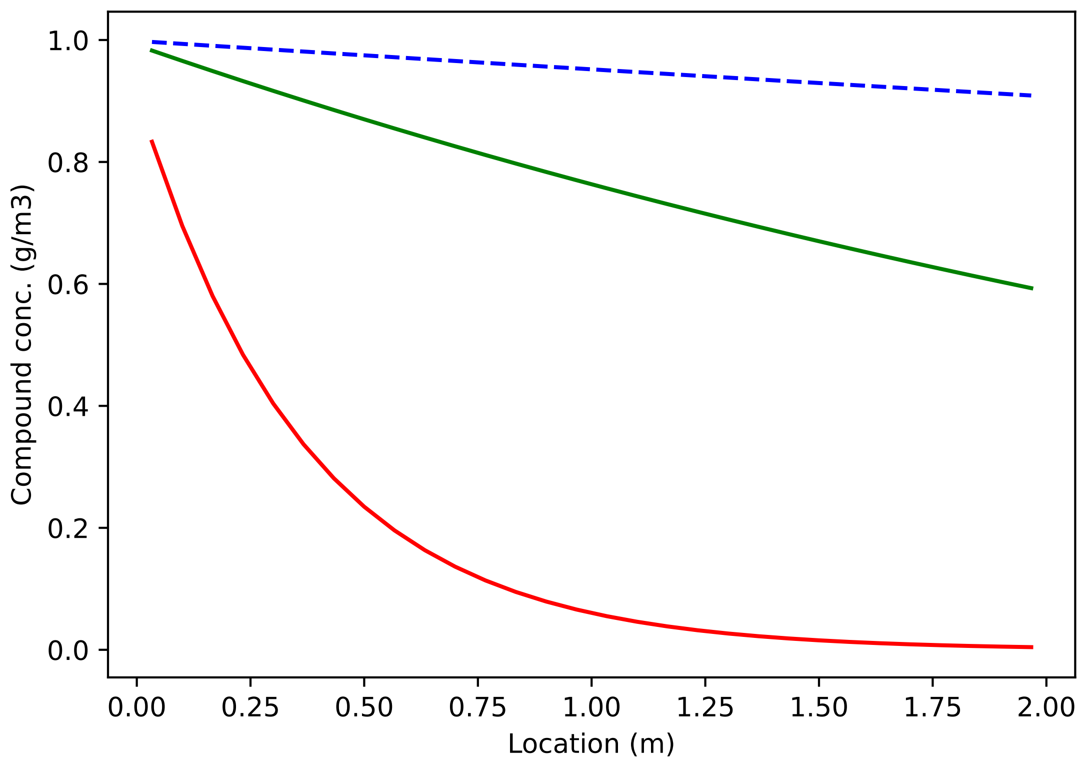
Liquid concentration (in last layer)

```
plt.plot(pred1[5] / 3600, pred1[1][nc - 1, :], 'r-')
plt.plot(pred2[5] / 3600, pred2[1][nc - 1, :], 'b', linestyle = 'dashed')
plt.plot(pred3[5] / 3600, pred3[1][nc - 1, :], 'g-')
plt.xlabel('Time (h)')
plt.ylabel('Compound conc. (g/m3)')
```



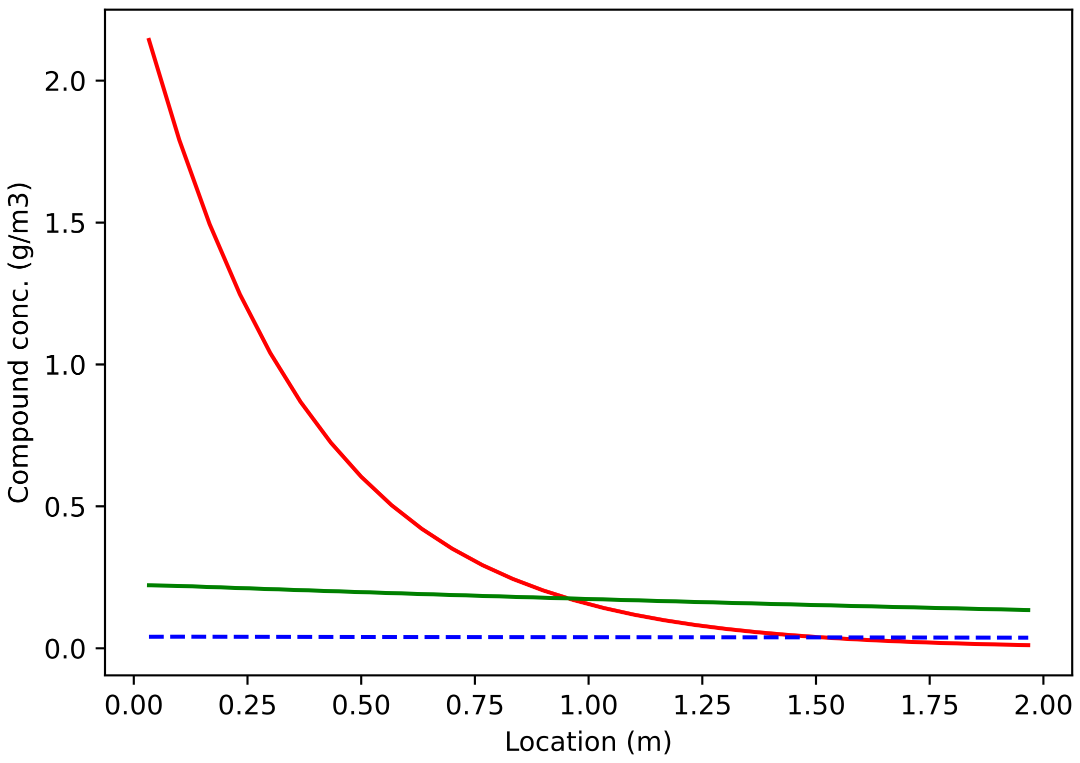
Profiles Gas

```
plt.plot(pred1[4], pred1[0][:, nt - 1], 'r-')
plt.plot(pred2[4], pred2[0][:, nt - 1], 'b', linestyle = 'dashed')
plt.plot(pred3[4], pred3[0][:, nt - 1], 'g-')
plt.xlabel('Location (m)')
plt.ylabel('Compound conc. (g/m3)')
```



Liquid

```
plt.plot(pred1[4], pred1[1][:, nt - 1], 'r-')
plt.plot(pred2[4], pred2[1][:, nt - 1], 'b', linestyle = 'dashed')
plt.plot(pred3[4], pred3[1][:, nt - 1], 'g-')
plt.xlabel('Location (m)')
plt.ylabel('Compound conc. (g/m3)')
```



Compound conc. (g/m3)
2.0
1.5
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Location (m)