Time-variable inlet concentrations

Sasha D. Hafner

09 June, 2023

Overview

Demonstration of time-variable inlet gas concentrations of target compound. This is done in R Markdown because there is no simple equivalent for Python and the reticulate package lets R do Python stuff.

R prep

Some R stuff.

```
library(reticulate)
```

```
## Warning: package 'reticulate' was built under R version 4.1.3
```

```
if(.Platform$OS.type == "windows") {
  use_python('C:\\Users\\sasha\\AppData\\Local\\Programs\\Python\\Python311')
} else {
  use_python('/usr/bin/python3')
}
```

And now the Python model

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 the notes in `tfmod.py` for more complete descriptions of inputs (and units).

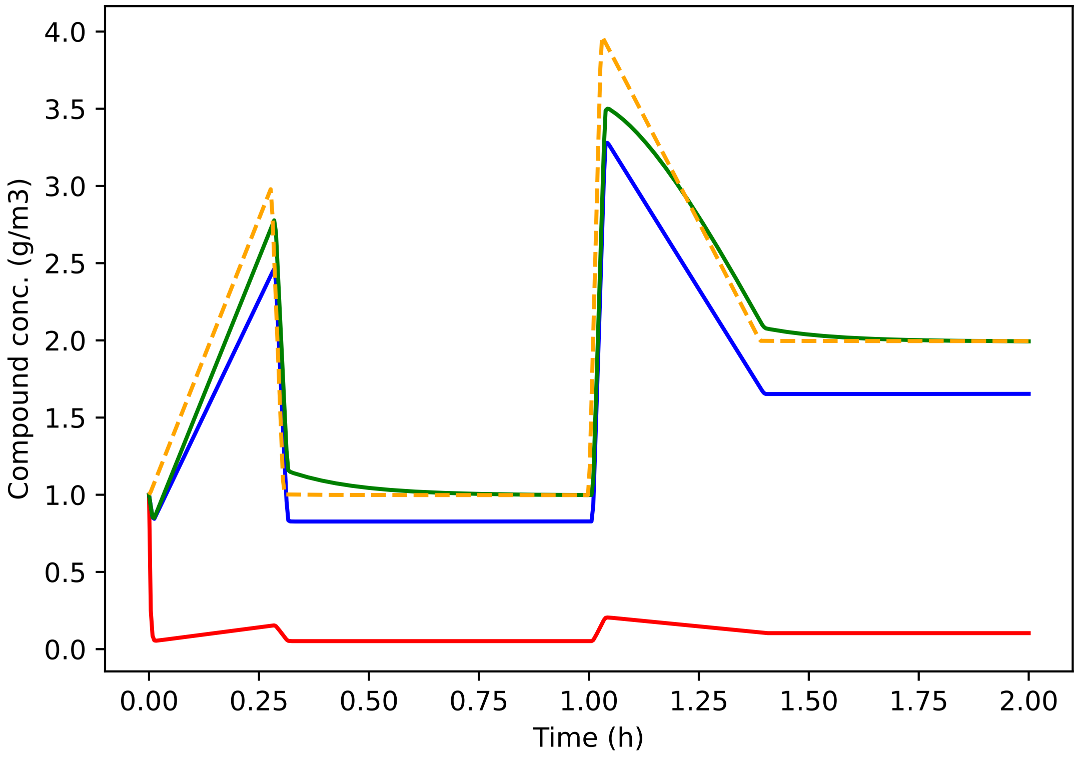
```
L = 2            # Filter length/depth (m)
por_g = 0.5      # (m3/m3)
por_l = 0.25     # (m3/m3)
v_g = 0.03
v_l = 2E-5
nc = 30          # Number of model cells (layers)
cg0 = 1          # (g/m3)
cl0 = 0          # (g/m3)
henry = (0.1, 2000.)
temp = 15.       # (degrees C)
dens_l = 1000    # Liquid density (kg/m3)

k = 500. / 3600  # Reaction rate (1/s)

pH = 7.
pKa = 7.

# Time-variable dirty air concentration coming in
#                  ~~~~~~~Time in seconds~~~~~~~~~~~~~~~    Concentration in g/m3
cgin = np.array([[0, 1000, 1100, 3600, 3700, 5000, 7200], [1, 3, 1, 1, 4, 2, 2]])
# Fixed for water
clin = 0.        # Fresh water concentration (g/m3)

# Times for model output, calculated from tt (total time) and nt (number of output times) here but coul
# Total duration (hours)
tt = 2
# Number of time rows
nt = 500
times = np.linspace(0, tt, nt) * 3600
```

Model scenarios

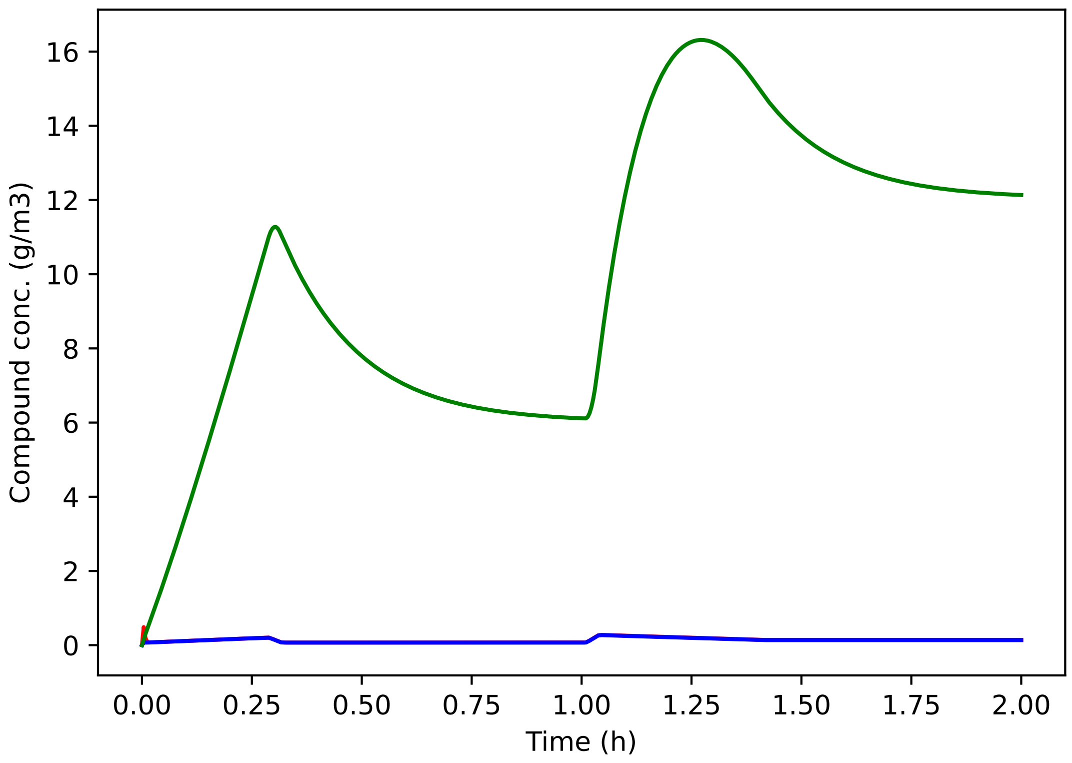
```
# Red line
Kga = 0.06
pred1 = tfmod(L = L, por_g = por_g, por_l = por_l, v_g = v_g, v_l = v_l, nc = nc, cg0 = cg0,
              cl0 = cl0, cgin = cgin, clin = clin, Kga = Kga, k = k, henry = henry, pKa = pKa,
              pH = pH, temp = temp, dens_l = dens_l, times = times)

# Onda correlation
# Blue line in plots
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)

# Turn off reaction to see concentration change
# Green line in plots
k = 0.
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)
```

Plots

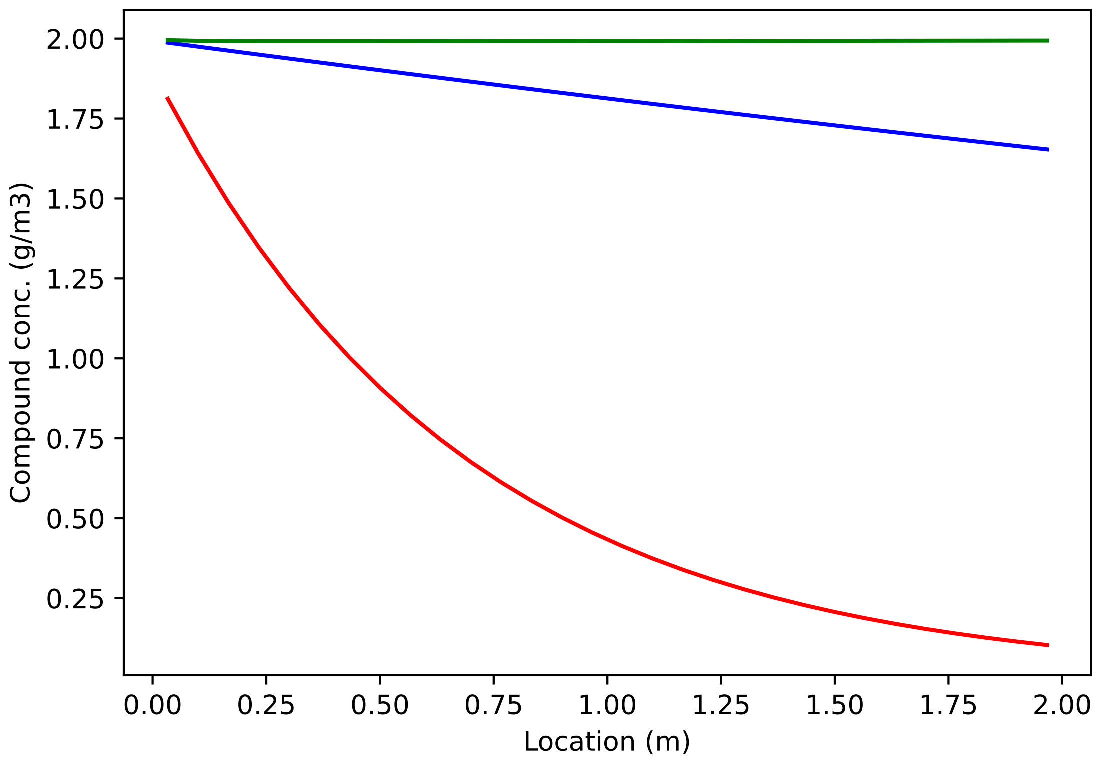
```
# Plot outlet concentration (= 1 - removal efficiency here because cgin = 1)
# Gas concentration (outlet air)
plt.plot(pred1[5] / 3600, pred1[0][nc - 1, :], 'r-')
plt.plot(pred2[5] / 3600, pred2[0][nc - 1, :], 'b')
plt.plot(pred3[5] / 3600, pred3[0][nc - 1, :], 'g-')
plt.plot(pred3[5] / 3600, pred3[0][0, :], 'orange', linestyle = 'dashed')
plt.xlabel('Time (h)')
plt.ylabel('Compound conc. (g/m3)')
```



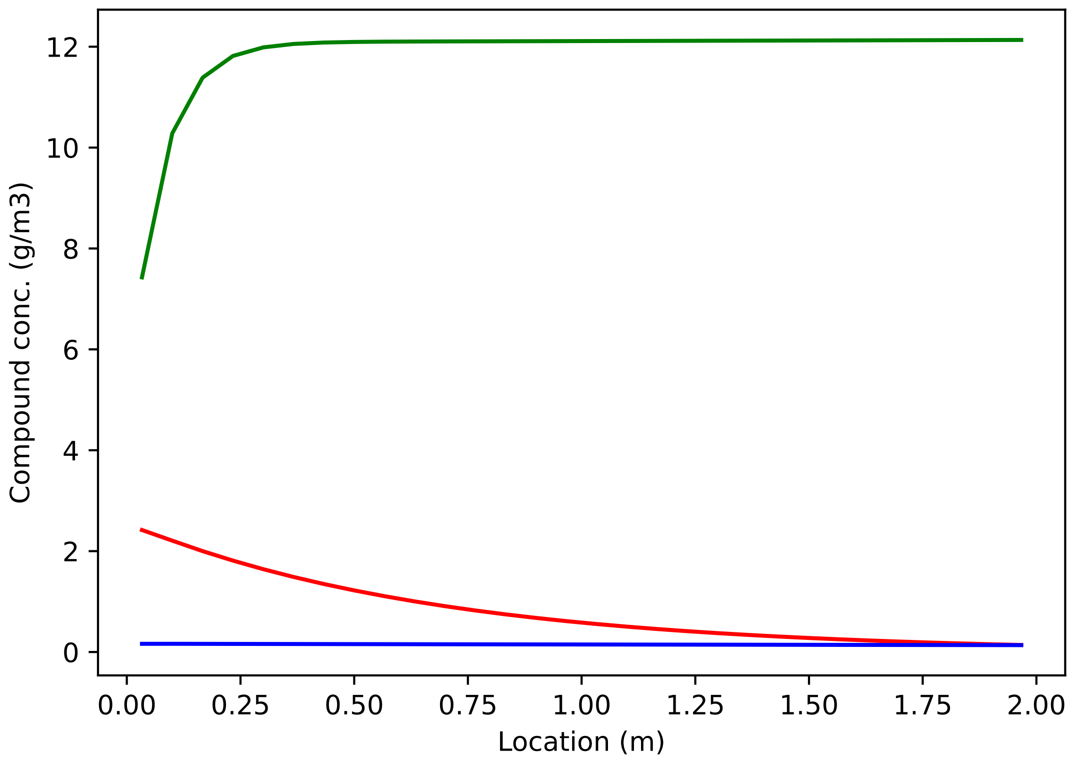
```
# Liquid concentration (in last layer)
plt.clf()
plt.plot(pred1[5] / 3600, pred1[1][nc - 1, :], 'r-')
plt.plot(pred2[5] / 3600, pred2[1][nc - 1, :], 'b')
plt.plot(pred3[5] / 3600, pred3[1][nc - 1, :], 'g-')
plt.xlabel('Time (h)')
plt.ylabel('Compound conc. (g/m3)')
```



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



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



Compound conc. (g/m3)
Location (m)
0
2
4
6
8
10
12
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00