

# **Quantitative pharmacology modeling (QSP) using Python – pqsp**

Kamaldeen Okuneye

This document support the use of the python code in  
<https://github.com/kamaldeen91/qsp-modeling-with-python>



October 28, 2019

# QSP modeling with python – pqsp

**pqsp** is a Python object oriented programming software used for simulating, analyzing and visualizing ODE models of quantitative systems biology and pharmacology. **pqsp** can be used to simulate the ODE models for different single and multiple drug dose regimen.

## Installation

This software requires Python 3.7 or previous versions, and the following prerequisite packages – numpy, pandas, and scipy. To install, first **clone** or **download** the repository. Depending on the computer OS: Windows – install the pqsp package using: **pip install -e /path/to/script/folder** in the command line OR navigate to the downloaded pqsp folder path in the command window and install pqsp using: **python setup.py install**. Mac OS – navigate to the downloaded pqsp folder path in the terminal window and install pqsp using: **python setup.py install** 🍷

## Getting started

To run a pharmacokinetic ODE model of your choice using any python IDE, follow example below:

```
❏ import pqsp

from pqsp.pqsp_single_dose_simulations import SingleDose
from pqsp.pqsp_multi_dose_simulations import MultipleDose
from pqsp.pqsp_multi_dose_delay_simulations import MultipleDoseDelay

from pqsp.pqsp_multi_bioav import MultipleDoseVaryBioav
from pqsp.pqsp_multi_bioav_delay import MultipleDoseVaryBioavDelay
```

```
❏ def my_model(y, t, ka, F, K, K12, K21):

    G=y[0]; A1 = y[1]; A2 = y[2]

    dGdt = -ka*G
    dA1dt = F*ka*G + (K21*A2 - K12*A1) - K * A1
    dA2dt = K12*A1 - K21*A2

    return [dGdt, dA1dt, dA2dt]

#####

ka = 1.8; F = 0.89; K = 0.28; K12 = 0.7; K21 = 0.3;
parameters = [ka, F, K, K12, K21]
```

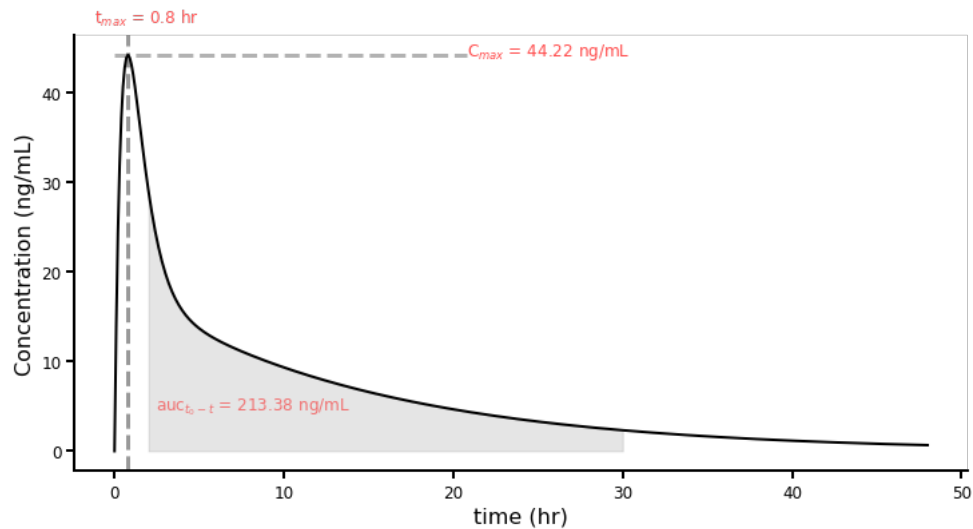
## Single dose simulations and plots

```
mymodel = SingleDose(my_model, parameters, number_of_compartments=3)

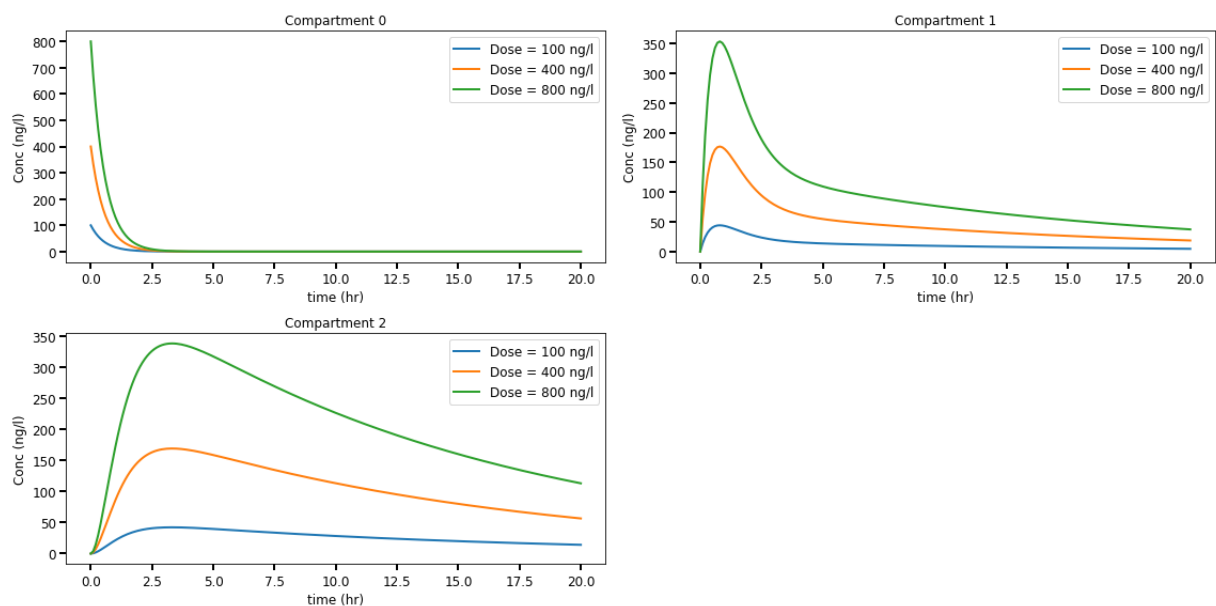
t0, C0 = mymodel.simulation(simulation_time=2, time_unit='day', dose_mg=[100], compartment_pos=[1])

mymodel.plot_simulation(t0, C0, show_max=True, show_auc=True, auc_start=2, auc_end=30)
```

Note that, although the model is a two-compartment model with oral administration, the number of compartments in the simulation is the number of ODE equations in the model.



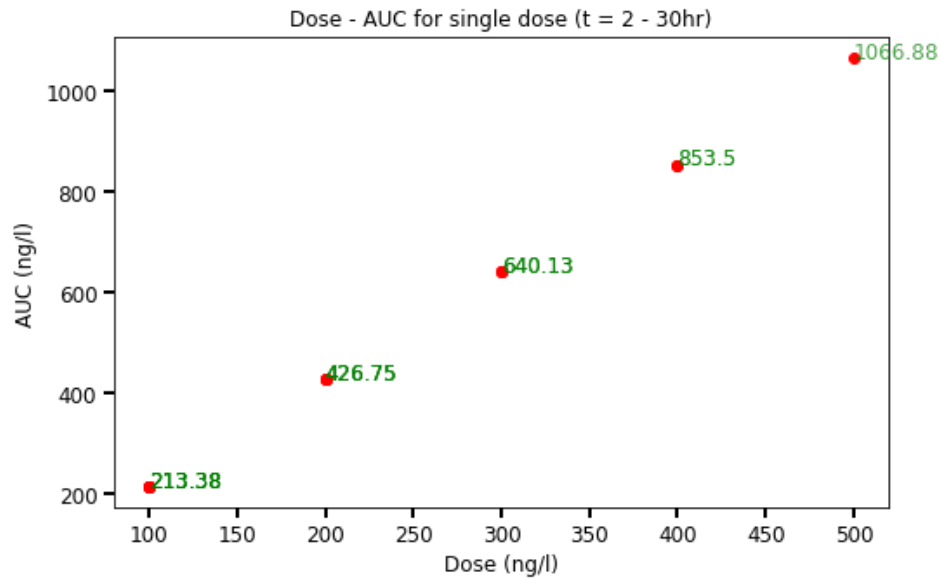
```
mymodel.single_dose_plot(simulation_time=20, time_unit='hrs', drug_doses=[100,400,800], compartment_pos=[0,1,2],
    figsize=(16,8))
```



```

mymodel.dose_auc_plot(simulation_time=3, time_unit='days', drug_doses=[100,200,300,400,500], compartment_pos=[1],
                      auc_start=2, auc_end=30, figsize=(8,5))

```



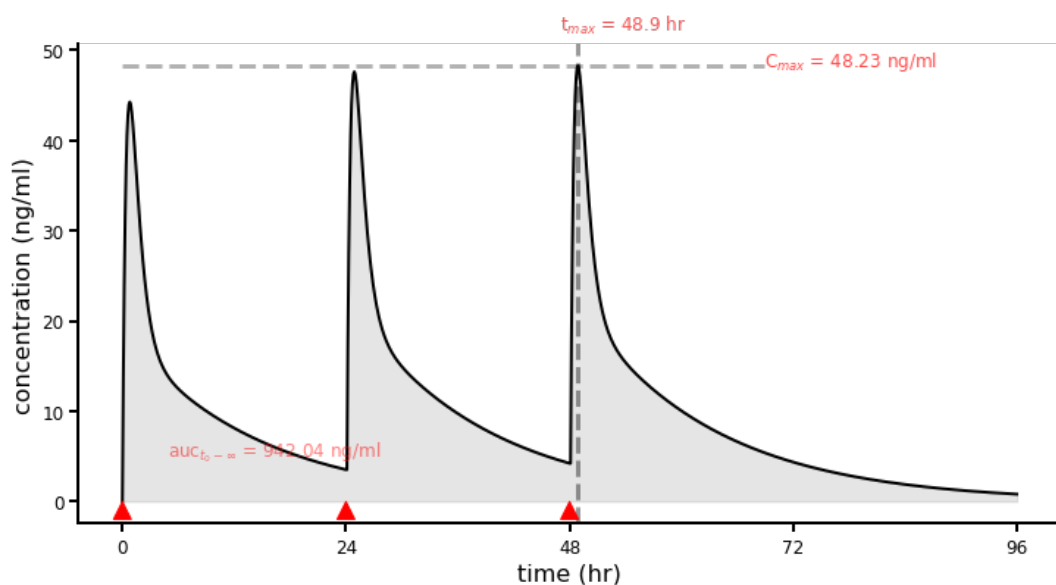
## Multiple dose simulations and plots

```

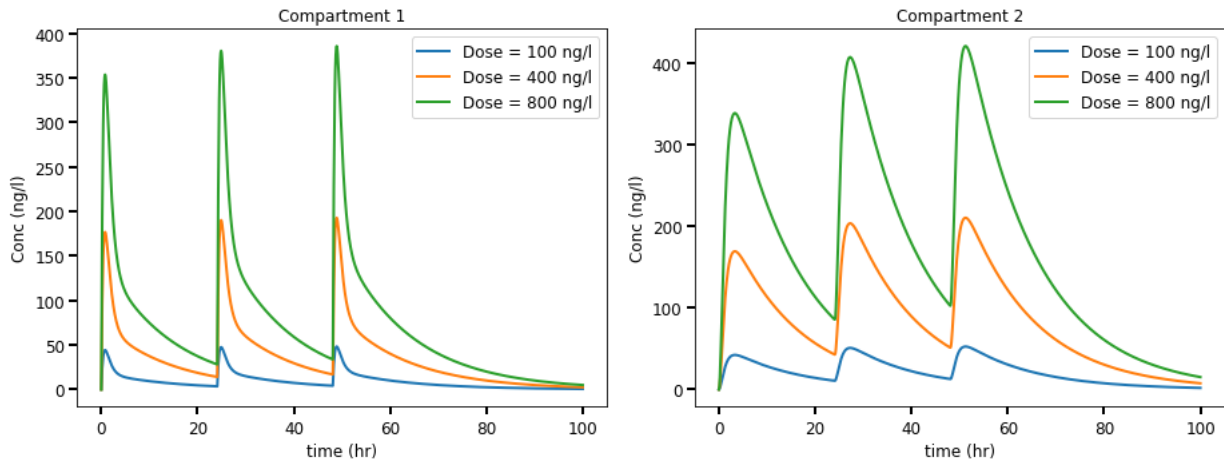
mymultimodel = MultipleDose(my_model, parameters, number_of_compartments=3, number_of_dose=3, interval=24)
time_1, conc_1 = mymultimodel.simulation(simulation_time=4, time_unit='days', dose_mg=[100], compartment_pos=[1])
# time_1, conc_1 = mymultimodel.simulation(simulation_time=10, time_unit='days', dose_mg=[100, 100, 100],
#                                           compartment_pos=[1])
mymultimodel.plot_simulation(time_1, conc_1, show_max=True, show_auc=True)

```

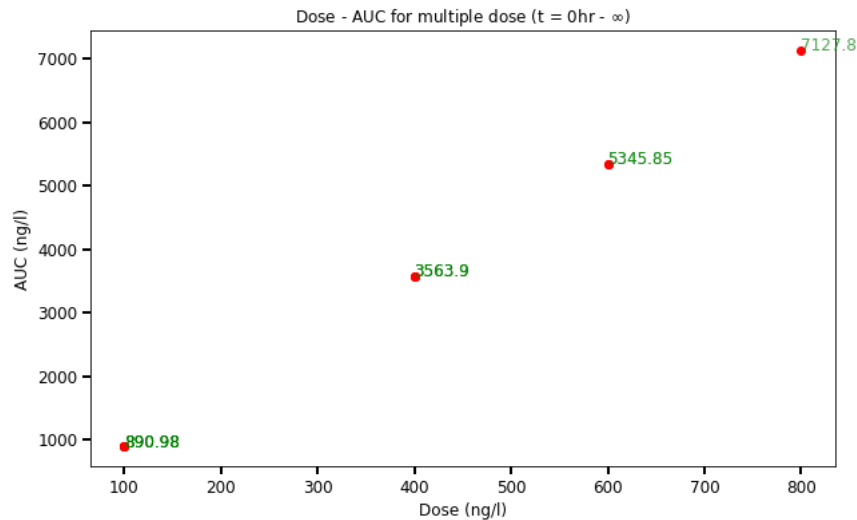
dose\_mg can also be written as dose\_mg = [100, 100, 100]. This allows for testing variation in dosage during therapy.



```
mymultimodel.multi_dose_plot(simulation_time=100, time_unit='hrs', drug_doses=[100,400,800], compartment_pos=[1,2],
                             figsize=(13,5))
```



```
mymultimodel.dose_auc_plot(simulation_time=3, time_unit='days', drug_doses=[100, 400, 600, 800], compartment_pos=[1])
```



When auc\_start and auc\_end are not specified, the simulation runs for 0 to  $\infty$ .

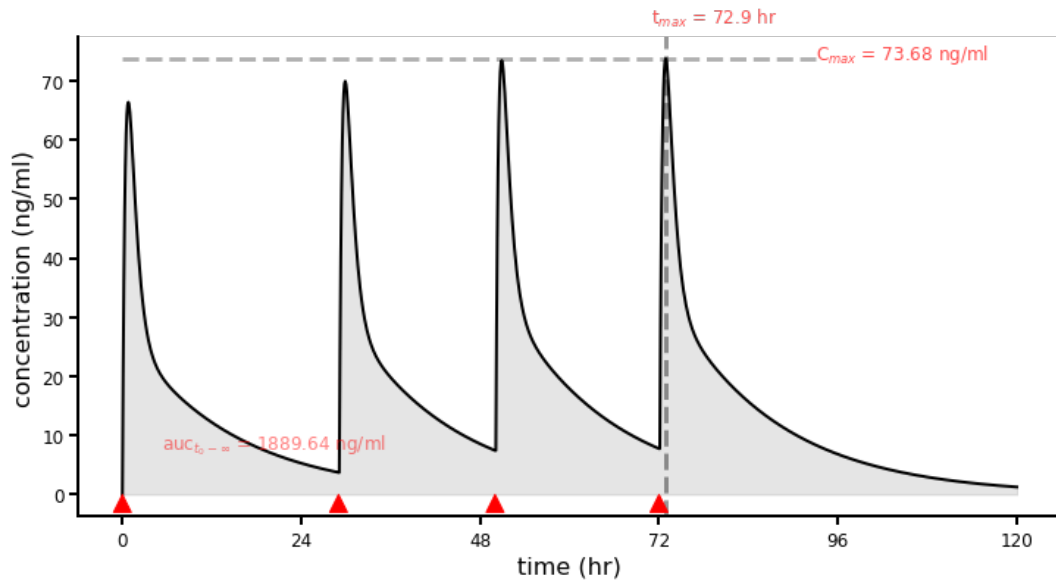
## Multiple dose with delay simulations and plots

```
mydelaymodel = MultipleDoseDelay(my_model, parameters, number_of_compartments=3, number_of_dose=4, interval=24,
                                 delay=[5, 2, 0])

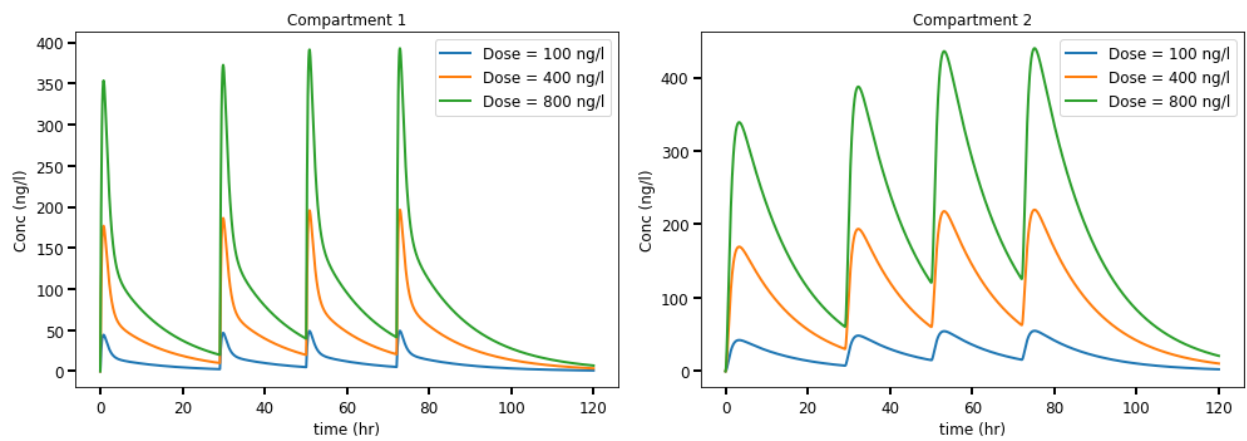
time_3, conc_3 = mydelaymodel.simulation(simulation_time=5, time_unit='days', dose_mg=[150], compartment_pos=[1])

# time, conc = mymodel_3.simulation(simulation_time=10, time_unit='days', dose_mg=[100, 150, 0, 100],
#                                   compartment_pos=[1])
mydelaymodel.plot_simulation(time_3, conc_3, show_max=True, show_auc=True)
```

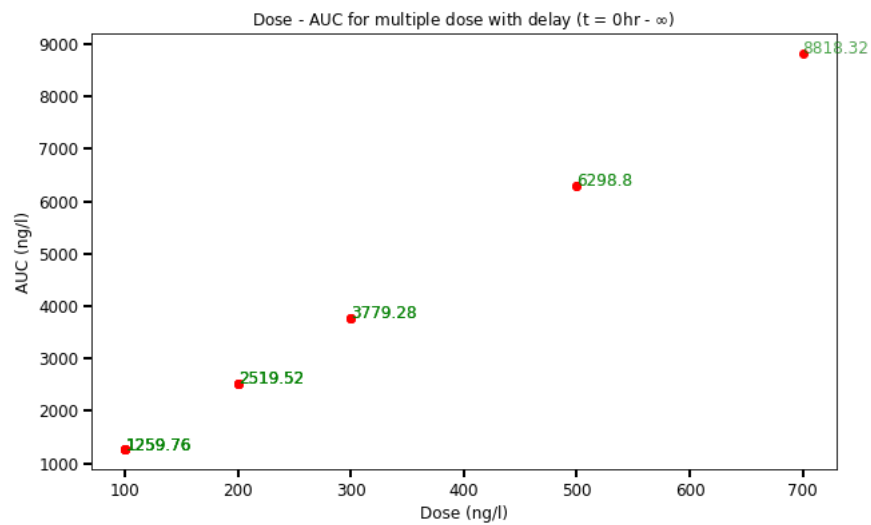
Note that number of entries in delay is one less than number of dose. This is because delay is assumed to occur in subsequent doses – after the first dose is taken.



```
mydelaymodel.multi_dose_delay_plot(simulation_time=5, time_unit='days', drug_doses=[100, 400, 800], compartment_pos=[1,2],
figsize=(14,5))
```



```
mydelaymodel.dose_auc_plot(simulation_time=5, time_unit='days', drug_doses=[100,200,300, 500, 700], compartment_pos=[1])
```



## Multiple with varying bioavailability simulations and plots

```

def two_c_model(y, t, ka, K, K12, K21, F):
    G = y[0]; A1 = y[1]; A2 = y[2]

    dGdt = -ka * G
    dA1dt = F * ka * G + K21 * A2 - K12 * A1 - K * A1
    dA2dt = K12 * A1 - K21 * A2

    return [dGdt, dA1dt, dA2dt]

ka = 0.17; Cl = 15.5; Vc = 368; Vd = 1060; Q = 16
K12 = Q / Vc; K21 = Q / Vd; K = Cl / Vc

par = [ka, K, K12, K21]

```

When defining model above, the bioavailability parameter  $F$  is listed as the last entry in the function. Always ensure this when executing the code for varied bioavailability.

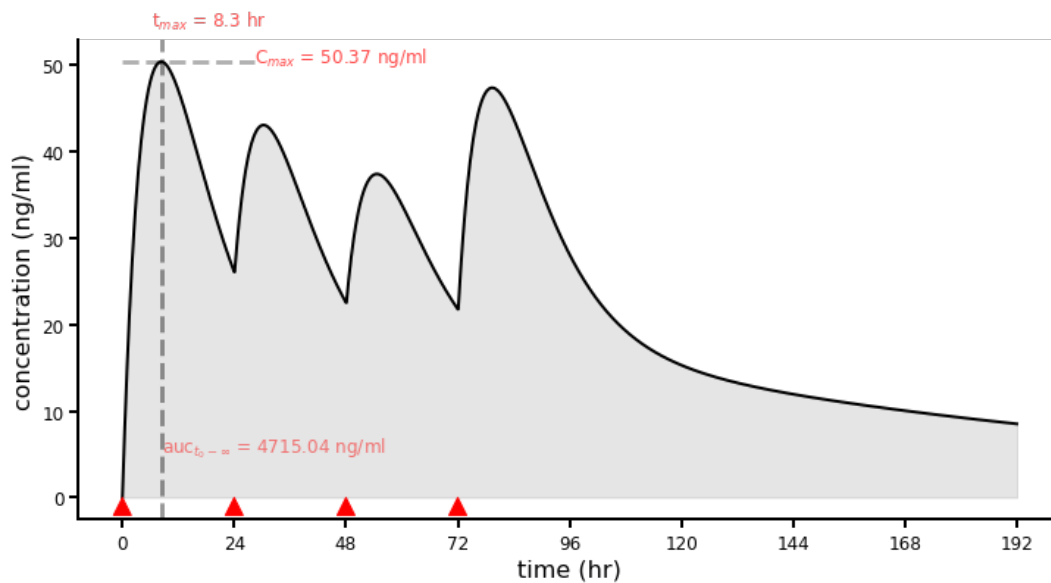
```

mymulti_biov = MultipleDoseVaryBioav(two_c_model, par, number_of_compartments=3, number_of_dose=4, interval=24,
                                     bioav=[1, 0.51, 0.41, 0.6])

time_3, conc_3 = mymulti_biov.simulation(simulation_time=8, time_unit='days', dose_mg=[100], compartment_pos=[1])

mymulti_biov.plot_simulation(time_3, conc_3, show_auc=True, show_max=True)

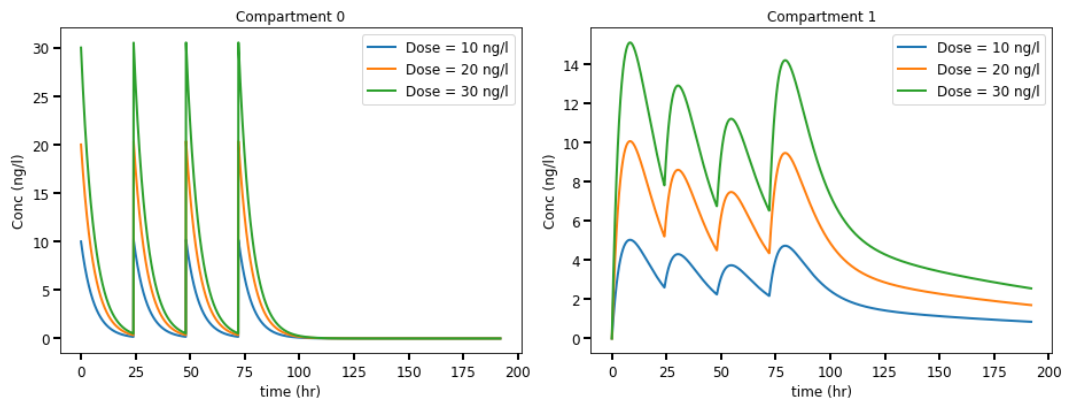
```



```

mymulti_biov.multi_dose_vary_bioav_plot(simulation_time=8, time_unit='days', drug_doses=[10,20,30], compartment_pos=[0,1],
                                         figsize=(13,5))

```



## Model with delay and varying bioavailability simulations and plots

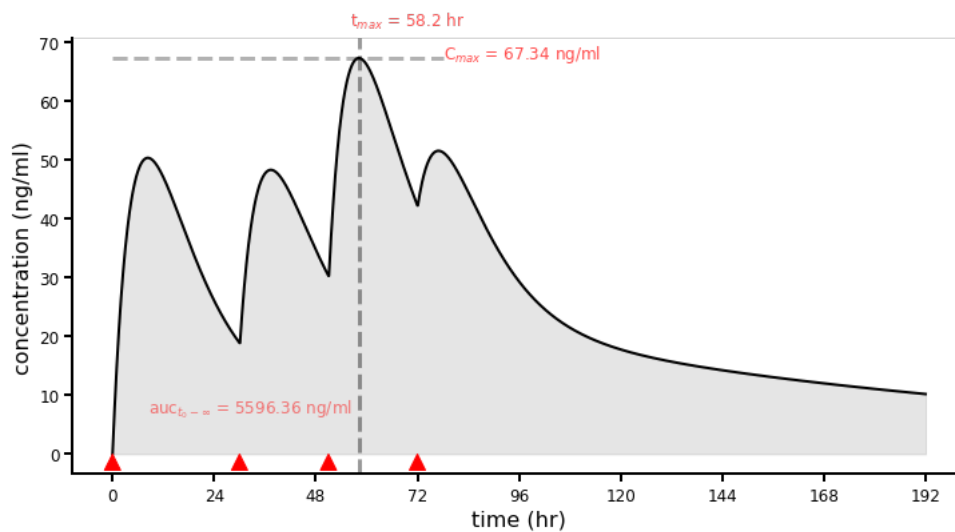
```

my_multi_biov_delay = MultipleDoseVaryBioavDelay(two_c_model, par, number_of_compartments=3, number_of_dose=4, interval=24,
delay=[6,3,0], bioav=[1, 0.7, 0.9, 0.4])

tnn, cnn = my_multi_biov_delay.simulation(simulation_time=8, time_unit='days', dose_mg=[100])

my_multi_biov_delay.plot_simulation(tnn, cnn, show_auc=True, show_max=True)

```

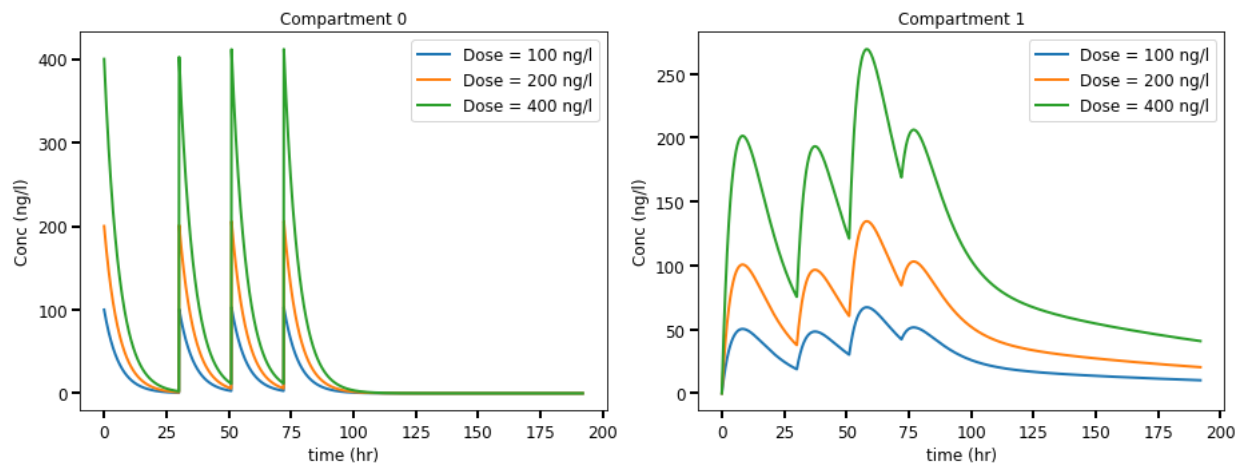


```

my_multi_biov_delay.md_delay_vary_bioav_plot(simulation_time=8, time_unit='days', drug_doses=[100, 200, 400],
compartment_pos=[0,1], figsize=(13,5))

```





## Try this

1. Run a single and multiple doses (with and without delay) simulations for an intravenous model given by

```
def my_intramodel(y, t, K, K12, K21):
    A1 = y[0]; A2 = y[1];
    dA1dt = (K21*A2 - K12*A1) - K * A1
    dA2dt = K12*A1 - K21*A2
    return [dA1dt, dA2dt]

#####
K = 0.28; K12 = 0.7; K21 = 0.3;
parameters = [K, K12, K21]
```

2. Run simulations for single, multiple doses with and without delay, and with varied bioavailability for any model(s) of your choice.
3. What are some of the errors you encountered when running the simulations and how did you correct them (eventually)?

