Python task

This task has two main focusses/ skills that it covers:

1. Automation of data analyses
2. Data fitting

These projects build upon concepts that we have covered. **You are strongly encouraged to utilise the internet to search for examples of the functions that you need to use**. However, you should make every effort to work out how to write these scripts.

In the previous Python tasks, we loaded and processed 220 data files. In this task, we're going to process 3960 files, and will require more sophisticated analyses, including function writing, and data fitting. Automating tedious processes is a strong point of coding! You would never do this analysis manually.

Please read this document in full before attempting the scripting.

Reminder/ explanation of our data

The data comes from a numerical model[1] of groundwater flow with contaminant transport. In this instance, there is water flowing from left to right. There is a contaminant that is released from the left-hand side of the model. Distributed throughout the model domain (Figure 1) are observation points where concentration over time 'data' were extracted from the model run.

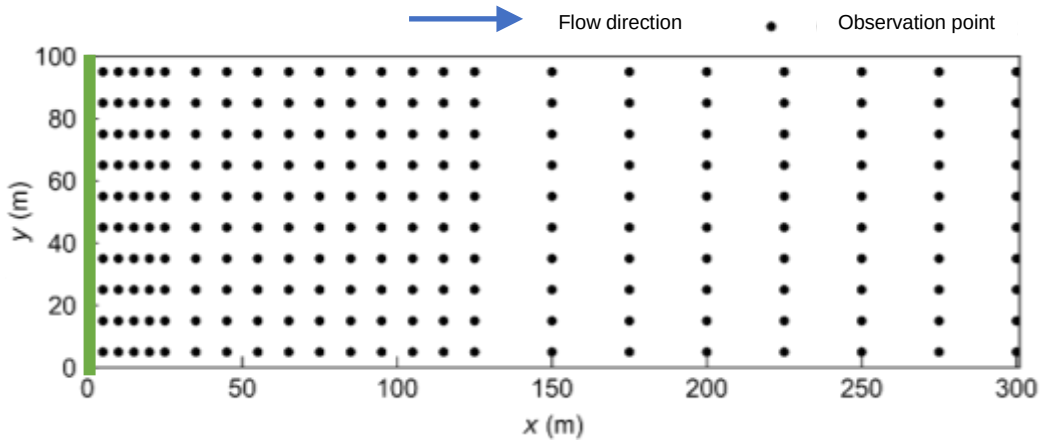

Figure 1: Model set up. A 'pollutant' (green line) is released from along the $x = 0$ boundary. Water flows from left to right. Black dots denote observation points where concentration over time data are written to output files.

The concentration of the hypothetical pollutant is 1 kg/m^3. We can use the concentrations over time to determine how fast the (ground)water is flowing in this area. At each of the observation points, we will see a 'break through curve' like Figure 2 (on the page below).

The moral to this story is that the time where the concentration first reaches half of the input concentration is related to the velocity. In fact, the equation that we are going to apply is:

$$v = x / t_{C>0.5}$$

[1] From the software HydroGeoSphere on the odd chance that you're interested.

i.e. the velocity, v (which is what we want to know) requires the distance from where our contaminant was released (x), as well as the time where the concentration first reaches 0.5 ($t_{C>0.5}$).

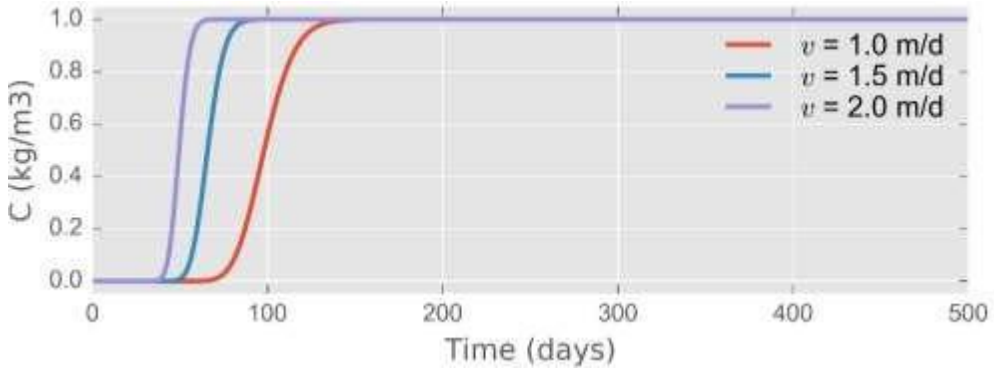

Figure 2: Hypothetical breakthrough curves for groundwater velocities of 1.0 m/d (red), 1.5 m/d (blue) and 2.0 m/d (purple). P.S. this is plot uses the ggplot style[2] (which is actually copied from R).

Extension of existing analyses (what is new to this task)

The concentration over time data (Figure 2, and in our data files) can be represented using the Ogata and Banks (1961) equation below. We will be writing a function to perform these calculations (over and over again).

$$C(x,t) = \frac{1}{2}\left[\operatorname{erfc}\left(\frac{x - vt}{2\sqrt{\beta v t}} \right) + \exp\left(\frac{vx}{\beta v} \right) \operatorname{erfc}\left(\frac{x + vt}{2\sqrt{\beta v t}} \right) \right]$$

Where C(x,t) is the concentration at a location x, and a time t, v is the velocity, and β is the parameter that we are looking for. For completeness, β is dispersivity, and has the units of length (i.e. m).

If you haven't seen it[3], erfc stands for the complementary error function. You don't need to worry too much about what it means. You will need to know that this is in the scipy.special library. So, to access it, you will need to load this. e.g.:

```
import scipy.special as sp

test = sp.erfc(1.0) # calculates the value erfc(1.0)
```

The exp function is a little more common. This is an exponential function. This is included in the numpy library, so to access it, you would write something like below:

[2] Look up style sheets here: https://matplotlib.org/gallery.html
[3] I assume that most haven't

```
import numpy as np

test = np.exp(1.0) # calculates the value exp(1.0)
```

What data are we working with?

Our previous task used a single model run. In this task, we are working with three different scenarios (labelled A, F and K), and for each scenario, there are 6 simulations.

Thus, in total, we have 3 (scenarios) × 6 (simulations) × 220 files = 3960 data files that we need to process.

I have edited these files so that they don't contain extra data that we don't require and reduced the number of decimal places (reducing the size of the files). The data are available as zipped files, by scenario. Once extracted, these files take up >900 MB, so you will need to work somewhere where you have sufficient space (i.e. USB).

Explanation of what our script is going to do…

Your script will need to include three functions:

1) – A function to apply the Ogata and Banks (1961) solution (see page 2, above)

Inputs: beta, x, v, t

Outputs: Concentration

2)- A root mean square error (RMSE) function

As the name suggests, this approach calculates the square root of the average error. Mathematically this is written as:

$$RMSE = \sqrt{\frac{\sum(obs - calc)^2}{n}},$$

where obs is the observed data (i.e. from the groundwater model), calc is what we calculate with our Ogata-Banks solution, n is how many values we have.

Looking at equations like these isn't for everyone. In fact, writing this code will be easier than you think. You simply calculate observed – calculated values (so you'll have an array of differences). You square these differences and calculate the average of the result. There are plenty of resources online to help you.

A high RMSE means that there is a large difference between the observed and calculated values. We want to search for the beta value that makes the RMSE as small as possible. Luckily, there's a function in Python to help us with this…

Your RMSE function should read in the relevant data from a file from the grounwdater model and calculate the Ogata-Banks solution. i.e. it should call your previous function[4].

Inputs: beta, x, v, t, observed concentration (with same times as in 1))

Output: RMSE

3- A function that minimises the RMSE by changing beta.

i.e. we want the beta that minimises the RMSE. This beta value is the goal of the analyses. We require access to a function in the scipy.optimize library called minimize. I suggest that you load the library as follows:

```
import scipy.optimize as opt

# and to call the function. NOTE, do not copy and paste this code, it won't work.

myresult = opt.minimize(the function name goes here, then the parameter that the function can change, then the arguments (called args), then other optional inputs)
```

Inputs: beta, x, v, t, observed concentration (with same times as in 1))

Output: optimised beta value.

So the goal of this third and final function is to call our previous two functions.

What do you need to hand up

- Your code as a PDF (to do this, from the File menu, select print, choose print to PDF)
  - Your code should be well commented, neat and tidy
  - You can hand up one or more .py files.
  - You should have a script that performs the analysis on either 1) the data from one folder at a time, 2) the data from one scenario at a time (i.e. 6 model runs), or 3) a file that processes all data in one go.
  - Your script(s) should produce the plots requested below
  - Your script(s) should also write summary file(s) of the analysis. i.e. the important information here are the scenario, the realisation, the x and y location, and the fitted beta values.
- A report that includes:
  - A discussion of the approach that you used in your analysis. i.e. did you loop by scenario, then realisation, then file (i.e. 3 loops), or did you run for each file (I hope not).
  - A plot of beta vs distance (x) for each of the Scenarios
  - A second plot, which builds on above which also includes the mean beta, and mean ± 2 standard deviations plotted.
  - You should include two examples of the concentration versus time from the groundwater model, and the calculated concentrations using a best fit beta value. Your plots should include the Scenario, realisation, file number and the fitted beta value. As an added challenge, can you also include the RMSE?
  - For those looking for an extra challenge, try investigating the method that Python uses in the fitting process. Also try changing some of the 'options'
- A list of all website addresses that you used to help you with your coding.

[4] This point is super important, as it will make your later coding much easier.

Please pay attention for:

- Quality, readability, how well commented, and how well written your code is. You can either have one or more scripts.
- How your tool runs. Is it efficient? How well automated is the process?
- The written report. Is it complete? Do figures have appropriate labels, are they referred to in the text?

There is no word or page limit on the report, there is also no proposed structure.

This task is primarily focussed on your code. The key point on the write up is to include (nice looking, appropriately labelled figures) from your analysis.

Tips to help you. Yes, these are hidden last. Hopefully you read this first.

Before you do anything, it would be a good idea to write a plan of how your code is going to work. i.e. how will your loops work, etc. Doing all of the thinking *before* you code is a really good idea.