**Applied Hydrogeology Practicals Weeks 8, 10 and 11**

For the remaining FloPy practicals in the course you will build on the model you created during previous FloPy practicals. All students will do PART A, which is described below in this handout. In this you will move from simulating flow in a simple homogeneous aquifer, to a model of a layered aquifer, and simulating the effect of pumping from a well pumping in the confined aquifer. You will then use this model to a) calculate source protection zones around the pumping well and b) consider the potential contamination of the well following a release of pollutants from a point source in the catchment. Note for some of these questions you will need to refer to material in forthcoming lectures on contaminant transport. Finally, in PART B – and this work is optional as explained below – you are offered the opportunity to raise your mark by choosing to develop your model and generate something original.

**PART A**: Remember – there is some code development from the model you have already generated – this mostly involves relatively simple developments which test your understanding of indexing. Having a fully working code at the end of this exercise is the minimum you will require to score 50% in this assessment. However, these practicals are critically not just about coding - you also need to be able to demonstrate a hydrogeological understanding by fully answering the questions set (highlighted in red in this handout). Whilst it is possible to get a first-class mark without answering the bonus questions these are potential extra marks available to.

Answers to questions in **BLUE** should be done as short comments to the code in the notebook, which will then be pushed to GitHub by 12.00 Wednesday 19th December.

Answers to questions in **RED** should be longer answers word processed in a separate document illustrated by screenshots. A hard copy of this should be handed to the School Office by 12.00 Wednesday 19th December, as well as being pushed to GitHub as part of your code.

All students are required to undertake the work in Part A (as described below), and a complete answer to this can gain you marks up to 75%. This part of the course will be supported by demonstrators who will be present in practical sessions in weeks 8, 10 and 11. Note that many of my research group will be at a conference from 17th December so we will not be in a good position to help you at the last minute!

And a final few points:

a) no help with the bonus question is available - you don't have to tackle these to be able to get a first class mark.

b) It is an individual assignment - its not an 'our code’ kind of group project - plagiarism is not tolerated nor will it stand you in good stead. Both code development and write-ups should be individual - as such we are happy to see you 1:1 to give you support

c) The demonstrators are not here to 'go through' your code outside the practical sessions - we will try to help you but require a clearly defined problem in your personal code, with evidence of personal independent effort.

Have some confidence in your own abilities, look through existing code and use the search tools as the first port of call to find answers to questions before running for help and remember to think about the hydrogeology as well as the coding.

**PART B** – Given the level of understanding that many of you have demonstrated in your previous FloPy work, we anticipate that the exercises in PART A should keep you occupied for two practical sessions. However, we have not assigned extra work for the final session. This is a deliberate change to last year and it recognises that for some the work set is challenging. Moreover, in order to encourage you to consider accessing the very highest marks (80% and above) we have created the opportunity for you to demonstrate your independent thinking and effort, commensurate with a research grade level performance.

To achieve a high first (>80%), and also be in with a chance (sound the trumpets!) to win the inaugural Excellence in Python based hydrological modelling prize awarded at the end of the year (complete with a small prize), you must not only show competency in the assigned work but develop it further. If this appeals then read on, but it is completely acceptable if you do not feel like you want to go down this route. There is no pressure to do this extra work; you should do it if you enjoy solving these kinds of problems or want to expand your understanding and experiences of FloPy/Python. The bonus extra up to 20% marks are very ‘expensive’ in terms of the effort you will need to put in to do a high-quality project. We will supply you with some research ideas, questions and snippets of code we have come across. If you are interested, then talk to Fiona about the research ideas – we will try to manage things to avoid people working on similar mini-projects. We will not support code development or bug fixing, but Fiona will discuss your ideas and look at the output of codes to assist in interpretations. Finished projects should be self-contained and will typically be openly shared online with the world. This is very much a research level experience!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PART A** – to be worked through at your own pace. Comprises the following subsections

PART A.1: Setting up the three-layered 3D model

PART A.2: Running tests using the layered 3D MODFLOW model

PART A.3: MODPATH

PART A.4: Running tests using MODPATH

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PART A.1: Setting up the three-layered 3D model**

Firstly, make a copy of the Unconfined Aquifer Practical from your previous FloPy practical ago and place in a new directory of your choosing.

**Add the following modules** (in addition to current ones – flopy and numpy);

import sys

import shutil

import matplotlib as mpl

import matplotlib.pyplot as plt #you can go ahead and remove the duplicate command underneath post processing of results – I’ve moved it here for asthetic reasons really

**Discretization**

**Start by changing the modelname to threelayer**

**Convert the model to;**

* 30 m thickness
* 3 layers of equal thickness

**At this stage both the time step and the Basic package parameters are fine, except that you should change the hydraulic head on the boundaries so that the hydraulic head at the site of the well before pumping is 5 m and the gradient across the model (h/l) is 2e-3. (This is done by changing the condeleft and conderight variables)**

**Layer-Property Flow package**

**For each of our 3 layers we want to define different aquifer properties.**

**Table 1 provides an initial estimate of rock properties for use in your modelling**

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Layer 1 | Layer 2 | Layer 3 |
| Horizontal Hydraulic conductivity (m/d) | 50. | 0.1 | 10. |
| Vertical Hydraulic conductivity (m/d) | 50. | 0.01 | 1. |
| Specific Yield | 0.05 | 0.05 | 0.05 |
| Specific Storage | 1.e-4 | 1.e-4 | 1.e-4 |
| Laytyp | Unconfined | Confined | Confined |

Use a list structure to define each variable. Hint: this is the same way we define variables like perlen

Lastly change the line that reads;

lpf = flopy.modflow.ModflowLpf(mf, hk=hk, vka=vka, sy=sy, ss=ss, laytyp=laytyp)

to;

lpf = flopy.modflow.ModflowLpf(mf, hk=hk, vka=vka, sy=sy, ss=ss, laytyp=laytyp**, ipakcb = 53**)

For those that want to know – **ipakcb = 53** is a flag to determine whether a cell budget file (a .cbc file )  is output after you run a modflow model. A cbc file is what is read into the MODPATH model for the particle tracking. By default according to FloPy’s manual it is set to 53 so you don’t need to set it as a parameter to the function but this flag has been a source of error in the past.

**Transient General-Head Boundary Package**

According to Darcy’s Law

Q= -KA(h1-h0)/(X1-X0)

Conductance combines the K, A and X terms so that Darcy’s Law can be expressed as

Q=-C(h1-h0)

**Qu 1a.** Derive and explain the dimensions of C (conductance)

The flow package calculates the conductance between cells using an average hydraulic conductivity of the cells, the area of the interface between cells and the distance between the cell centres. Some of the head-dependent boundary conditions – including the General-Head Boundary - require the user to specify the conductance.

The General-Head Boundary package is the simplest of several head-dependent flux boundary packages.  In it, there is a linear relationship between the ground water flux into (or out of) the cell and the head in the cell. The user specifies a reference head and a conductance.  When the head in the cell equals the reference head, the flux is zero.  If the head in the cell is less than the reference head, water enters the groundwater system through the general-head boundary.  If the head in the cell is greater than the reference head, water leaves the groundwater system.

If you look in the first *for* loop you will see the hk variable referenced beside condleft and condright. Given we now have 3 different values for hydraulic conductivity we need to create the correct boundary condition for each cell depending on its hydraulic conductivity.

**Use hk[XXX] in place of hk where XXX takes the value of the iteration variable.**

Tip- Insert **print ()** into various stages of for loops to get an idea of intermediary variables of hk at different stages of the for look.

**Divide zbot by 3 – Why? Answer as an adjacent comment.**

**Under the section** *#Make list for stress period 2***, delete the lines containing condleft and condright.**

**Copy the condleft and condright lines you just edited from the first ‘for il in range(nlay):’ into a similar place for the second ‘for il in range(nlay):’ loop.**

**Transient well package**

Currently the well is intersecting the 0’th layer (i.e. the 1st model layer – remember indexing in Python indexing starts at 0 – you just convinced yourself of this by looking at how hk[1] works)

**Change this so that it pumps out of the 3rd layer at a rate of 1.157 L/s (remember to use consistent units in your modelling – HINT the current value is 100 m3/day).**

**Output Control, Preconditioned Conjugate Gradient Packages, Writing the MODFLOW Data Files, Running the Modelling – Leave the same**

**Post-Processing the results**

You can delete import matplotlib.pyplot as plt as we already included it above in the initial set of imports. It does no harm here, but the advantage of moving this to the top of the script is that we will shortly be using it in the MODPATH simulations.

Leave import flopy.utils.binaryfile as is. We will only use it in this cell. That’s the aesthetic rule of thumb; use something once - import close to the function, use it many times - import at the start.

At some point you may decide to alter the length of time we allow the simulation to run to (or you may not!). At present;

mytimes = [1.0, 101.0, 201.0]

This references the final timesteps at each of the three stress periods. Print out the variable times and see for yourself that the final timestep is 201.0

If we choose to change the length of the simulation we will need to change 201.0. **Using the times variable and indexing feed the last timestep (for a simulation of any length) into the mytimes variable in the place of XXX;**

mytimes = [1.0, 101.0, XXX]

**In the lines;**

plt.imshow(head[0, :, :], extent=extent, cmap='BrBG', vmin=0., vmax=10.)

plt.colorbar() # plot colourbar

CS = plt.contour(np.flipud(head[0, :, :]), levels=levels, extent=extent,    zorder=10)

head[0, : , :] refers to all rows and all columns of the 0th (i.e. 1st) layer.

**You can change the value of 0 to see the values of heads in different layers**. This may come in handy when you want to view the effect of pumping on the bottom aquifer.

**Move all the code under # Plot the head versus time (i.e. from idx = (0, nrow/2 - 1, ncol/2 - 1) to plt.show()) to a new cell bellow.** At present this code only plots the data for the first layer. We are going to change this so that it plots data for the same row/column but for each layer.

**Add 2 new idx variables which reference layers 2 and 3 (hint in python this would be the 1st and 2nd) Comment on the relationship between the well cell and the cells described by the idx variables adjacent to code.**

**Add 2 new ts variables using the new idx variables**

**What does the get\_ts function do? Comment adjacent to code.**

**Plot those two new variables on top of the existing head**

**Use the plt.plot(… label=’something’) argument to give each line a name**

**Use plt.legend() to display those labels on the graph.**

You should now see three lines and a legend on the plot.

**Qu 1b. Using the figure you generate- briefly (3-4 sentences) comment on the differences between these lines (HINT: Try changing vka in the middle (aquiclude) layer to get a clearer idea of what’s going on and the reasons for the differences).**

**Run the function get\_kstpkper() on the headobj variable**. **What does this function do? Comment adjacent to code.** Go to the flopy documentation and type ‘get\_kstpkper’ into search. If you put the () in the search you won’t find it.

We wish to see the head data for the last time step of the last stress period. This is because we want to see the steady state head situation. In fact actually because we specified the second stress period to run to a steady state situation (steady=True) the very first time step of the second stress period is at steady state. See for yourself how changing the third value for the steady variable from True to False affects the model. **Paste the following into a cell;**

**hds = headobj.get\_data(kstpkper=(99, 2))**

Check in the documentation to see how this get\_data function works. Print out the hds variable and reference with the perlen and nstp variables so you can assure yourself you understand how everything works.

If you print hds[:,:,:] you will see that that it’s a numpy array. It’s a 3D numpy array of 3x10x10 dimensions – see for yourself by running; np.shape(hds).

It has a 3x10x10 shape at this stage. The command hds[:,:,:] calls for python to read out **all** (i.e. using a **colon**  **:** ) the layers (0th **:** ), rows (1st **:** ) and columns (2nd **:** ).

So if we were to call hds[1,3,5] this would give us a single value corresponding to a cell in the 2nd layer, 4th row and 6th column.

**Adapt the following code to**;

* Generate cross-sections showing variation of head in both x and y plane – change the 2nd and 3rd indexing values (HINT – Only 1 index value needed)
* View distribution of head in the different layers – change the 1st indexing value

**plt.imshow(hds[x, x, x])**

**plt.colorbar();**

**Adapt the following code to**;

* Generate cross-sections of head values in various layers (HINT – You need 2 indexes)

**plt.plot(hds[x, x, x])**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PART A.2: Running tests using the layered 3D MODFLOW model**

Start by increasing the spatial resolution of the model – **change nrow and ncol to 30**. This should help us see the effects of the tests we are about to run better.

**BONUS POINTS (FOR THE COMPUTATIONALLY CURIOUS – answer in word document as BONUS 1):** What happens to the speed of the code, specifically the running of the executable, when we increase the resolution of the model? Plot your results. Talk to demonstrator if interested at looking at this problem.

**Change the pumping rate in stages from 1.157 L/s to 34.7 L/s**

**Q2a -** Can you see a problem evaluating the effect of pumping at the higher rate using this model? Write an explanation of this problem in the word document and include suitable plots as supporting evidence**.**

**Given your observations and interpretations in Q2a change the model dimensions. Remember to maintain a hydraulic gradient of 2e-3 and keep the hydraulic head of the well cell at 5m. MODFLOW does not let you set negative head boundary conditions.**

**Q2b –** Explain with the aid of appropriate diagrams how changing the volume of the aquifer simulated has improved the model.

**Model validation**

K values for Layers 1 and 3 given in Table 1 are well constrained, derived from the geometric mean of a large number of pumping tests in these aquifers. K values for the aquiclude (Layer 2) given in Table 1 are the geometric mean of those derived from core plug laboratory tests as no pumping test values are available.

**Q3a.** Evaluate your model by attempting to simulate head measured in piezometers within Layers 1 and 3 (data in Table 2) at steady state when the well is pumped at **34.72 L/s**. How well do the simulated heads in the model using parameter values in Table 1 match the observed values? Quantify the degree of fit using the root mean squared error (RMSE)

where n is the number of observations, the predicted value predicted by the model and *yj* the observed value.

**Q3b.** Adjust values of hydraulic conductivites (kh and kva) for the aquiclude (layer 2) in your model to generate the best match. Aim for an RMSE of 0.1m between your predicted heads and the observed heads reported in Table 3. Discuss sensitivity of the predicted cone of depression in both aquifers to these changes and present your best fit model. Suggest reasons the core plug measurements of hydraulic conductivity for the aquiclude may have provided a poor fit to the observed drawdown.

Table 2 –Head measured in piezometers within the upper (layer 1) and lower (layer 3) aquifers at steady state pumping of 34.7 L/s. Positions of observation wells are given as distances in m relative to the well. Positive values are distances to the west and north of the well, and negative values are distances to the east and south of the well.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer | Piezometer | X (m) | Y (m) | Head (m) |
| 1 | A | 834 | 0 | 5.80 |
| 1 | B | 0 | -667 | 3.82 |
| 1 | C | 0 | 0 | 1.60 |
| 1 | D | -167 | 0 | 2.38 |
| 1 | E | 0 | 333 | 3.28 |
| 3 | F | 0 | 0 | 1.01 |
| 3 | G | 0 | 167 | 2.73 |
| 3 | H | -500 | 0 | 2.59 |
| 3 | I | 0 | 333 | 3.28 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PART A.3 MODPATH**

MODPATH is a post-processing model that uses the output from the flow simulator to compute 3D the pathway of individual particles. If you give an initial, starting position for the particle, then you can use numerical techniques to track the particle in 3D over a series of discrete time steps. A particle's path is tracked from one cell to the next until it reaches a boundary, an internal sink/source, or satisfies some other termination criterion. Or you can specify an end point and MODPATH can be used to track a particle back to the source.

Particle tracking helps you to visualize a flow field, detect errors in the flow model and check mass balances for individual cells or for regions. Applications of particle tracking include:

* capture zones in order to specify wellhead protection areas, and calculate strategies for hydraulic control of contaminant plumes
* time of travel to calculate arrival of contaminants, time to clean up and to estimate recharge
* model calibration by comparison with isotope data on groundwater ages.

MODPATH uses a semi-analytical solution to this equation and linear interpolation between point data. We only know head at specific points at specific times; linear interpolation is used to obtain velocity at other points and times.

Advantages of particle tracking:

1. Can determine pathlines in spatially variable parameter domains
2. Can determine three-dimensional pathlines
3. Compute forward and reverse-tracked pathlines
4. Confine, leaky, and unconfined aquifers

It is important to remember that pathlines are not flow nets - they are not solutions to the Laplace equation and there is not equal amounts of flow between individual path lines.

**Q4.** Write a short paragraph giving several reasons why travel times calculated using particle tracking might be shorter than those derived from a field-scale tracer test carried out using an unreactive tracer.

MODPATH input data consists of a combination of MODFLOW input data files, MODFLOW head and flow output files, and other input files specific to MODPATH. MODPATH produces several output files, including a number of particle coordinate output files. These can be used as input data for other programs that process, analyze, and display the results in various ways.

We will need a new executable, the MODPATH engine which is called called mp6, to run these simulations. The processes getting this executable are the same as when we got the first mf2005 executable.

**MAC – If these instructions do not work for you please report difficulties**

Try <https://water.usgs.gov/ogw/modpath/>

Download **Version 6** (listed under additional resources) – **DO NOT download the latest release version 7**

Copy the file mp6 to a folder somewhere locally. It is best to put it a folder in your applications folder or perhaps inside of a folder inside of the modflow folder

If that doesn’t work – you need pymake/modflow

<https://github.com/modflowpy/pymake>

It’s a collection of python scripts that download and install modflow executables for the mac – pretty useful.

**cd** into the pymake directory (wherever you cloned it to last time)

**cd examples**

**python make\_modpath6.py**

This then creates the mp6 executable in the temp folder. Move it to the place (folder in Applications most likely) where you stored mf2005.

**Windows – If these instructions do not work for you please report difficulties**

<https://water.usgs.gov/ogw/modpath/>

Download **Version 6** (listed under additional resources) – **DO NOT download the latest release version 7**

Go to the bin

Copy the file mp6 to a folder somewhere locally. I’d recommend putting it a folder in your applications folder or perhaps inside of a folder inside of the modflow folder

**MODPATH Model Creation**

In a new cell **add the following code;**

mp = flopy.modpath.Modpath(modelname='Modflowmod', exe\_name='/YOUR/PATH/TO/mp6',

modflowmodel=mf,

dis\_file='threelayer.dis',

head\_file='threelayer.hds',

budget\_file='threelayer.cbc')

In a new cell bellow, convert to markdown (click on left hand side of cell till it turns blue and press ‘m’), and briefly describe what each of the of the arguments is doing. Try and format your answer using headings (#) so that it is easily readable.

In a new cell **add the following code;**

mpb = flopy.modpath.ModpathBas(mp, hdry=mf.lpf.hdry, laytyp=mf.lpf.laytyp, ibound=1, prsity=0.05) #note prsity = effective porosity

sim = mp.create\_mpsim(trackdir='forward', simtype='pathline', packages='RCH', start\_time=(2, 0, 99.))

mp.write\_input()

mp.run\_model(silent=True)

In a new cell below, convert to markdown, and briefly describe how the arguments to ModpathBas and create\_mpsim are working.

In a new cell **add the following code;**

epobj = flopy.utils.EndpointFile('ModflowMod.mpend')

well\_epd = epobj.get\_destination\_endpoint\_data(dest\_cells=[**(X, X, X)])**

Input an appropriate location into the values for **X**

Print well\_epd - If it is blank you probably have done something wrong!

In a new cell **add the following code;**

fig = plt.figure(figsize=(14, 14))

ax = fig.add\_subplot(1, 1, 1, aspect='equal')

modelmap = flopy.plot.ModelMap(model=mf, layer=2)

quadmesh = modelmap.plot\_ibound()

linecollection = modelmap.plot\_grid()

quadmesh = modelmap.plot\_bc('WEL', kper=2, plotAll=True)

contour\_set = modelmap.contour\_array(hds,

levels=np.arange(np.min(hds),np.max(hds),0.5), colors='b')

plt.clabel(contour\_set, inline=1, fontsize=14)

modelmap.plot\_endpoint(well\_epd, direction='starting', colorbar=True)

We can also add particle pathlines

In a new cell **add the following code;**

pthobj = flopy.utils.PathlineFile('ModflowMod.mppth')

well\_pathlines = pthobj.get\_destination\_pathline\_data(dest\_cells=[(2, 14, 14)])

In a new cell **add the following code;**

fig = plt.figure(figsize=(14, 14))

ax = fig.add\_subplot(1, 1, 1, aspect='equal')

modelmap = flopy.plot.ModelMap(model=mf, layer=0)

quadmesh = modelmap.plot\_ibound()

linecollection = modelmap.plot\_grid()

quadmesh = modelmap.plot\_bc('WEL', kper=1, plotAll=True)

#contour\_set = modelmap.contour\_array(hds,

# levels=np.arange(np.min(hds),np.max(hds),0.5), colors='b')

plt.clabel(contour\_set, inline=1, fontsize=14)

modelmap.plot\_endpoint(well\_epd, direction='starting', colorbar=True)

for pid in np.unique(well\_pathlines.particleid):

modelmap.plot\_pathline(pthobj.get\_data(pid), layer='all', colors='red');

Finally, we can define areas by travel time (Note: by default MODFLOW considers time in days not the SI unit of seconds!).

In a new cell **add the following code;**

fig = plt.figure(figsize=(14, 14))

ax = fig.add\_subplot(1, 1, 1, aspect='equal')

modelmap = flopy.plot.ModelMap(model=mf, layer=2)

quadmesh = modelmap.plot\_ibound()

linecollection = modelmap.plot\_grid()

quadmesh = modelmap.plot\_bc('WEL', kper=2, plotAll=True)

modelmap.plot\_endpoint(well\_epd, direction='starting', colorbar=True)

a=np.full(hds.shape, np.nan)

for i in well\_epd:

a[i[6],i[7],i[8]]=i[4]

travtime\_set=modelmap.contour\_array(a[0],levels=np.arange(0,1000,100))

plt.clabel(travtime\_set, inline=1, fontsize=10)

You can change the arguments to np.arange of 100 to derive contours with greater or smaller contour intervals and 1000 to expand the max value if necessary. I.e. its np.arange(start,stop,interval). np.arange is generating an numpy array. You can also develop a list in place of np.arange() if you would like a truly custom contour solution.

The Environment Agency is interested defining source protection zones (SPZs) based on groundwater travel times.

**Zone 1** is closest to the well showing the area of highest risk and defined by 50-day travel time from any pollution below the water table to the groundwater source (minimum for decay of some biological contaminants).

**Zone 2** is defined by a 400-day travel time (minimum recharge area required to support 25% of the protected yield).

**Zone 3** is the area within which all groundwater recharge (from precipitation or surface water) is presumed to be discharged at the well - covers the whole of the catchment area of a well based on the area needed to maintain abstraction, assuming that all water will eventually reach the abstraction point. Designed to avoid the long-term degradation of the quality of water in the aquifer.

**Q5.** For your best fit model, generate a map of the source protection zones 1, 2 and 3 around your well using a porosity of 0.05. Explain and illustrate the sensitivity of travel time to the specified porosity.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PART A.4: Running tests using MODPATH**

After 10 years of pumping good quality water from the well, an underground storage tank full of inorganic contaminant E-VIL located 1500 m due west (up-hydraulic-gradient) of the pumping well started to leak. The base of the storage tank sits immediately above the water table. As the contaminant passes through the groundwater system a range of different chemical and biochemical processes act to transform, retard, and attenuate solutes. Sorption, which controls the distribution of mass between the solid and solution phase, was determined in the laboratory using rock material from all three aquifer layers. Experimental data is summarised in Table 3.

Table 3. Experimental data on sorbtion of the contaminant which was spilled onto a

|  |  |  |
| --- | --- | --- |
| Aquifer Layer | Equilibrium concentration of E-VIL mg/L | Contaminant (E-VIL) sorbed ug/g |
| 1 | 0.08 | 13 |
| 1 | 0.10 | 17 |
| 1 | 0.27 | 46 |
| 1 | 0.40 | 67 |
| 2 | 0.53 | 86 |
| 2 | 0.67 | 105 |
| 3 | 0.20 | 31 |
| 3 | 0.32 | 51 |

**Q6a.** Plot and describe the form of the isotherm and calculate the distribution coefficient Kd (L/kg) from the mass of solute sorbed/dry weight of solid and the concentration of solute in equilibrium with mass of solute sorbed on solid.

**Q6b** Use the distribution coefficient derived from the experimental data in Table 3 to calculate the retardation factor, *rf*

Where

d = bulk density of the aquifer material is 1600 mg/cm3  
 = porosity   
Kd = partition coefficient

**Q6c.** Using your retardation factor (Rf ) and the equation below, estimate how long the contaminant will take to reach the pumping well. When would you start monitoring the well for the appearance of E-VIL? Justify your recommendation.

Rf = Vp/Vc

Where

Vp = velocity of water through the control volume

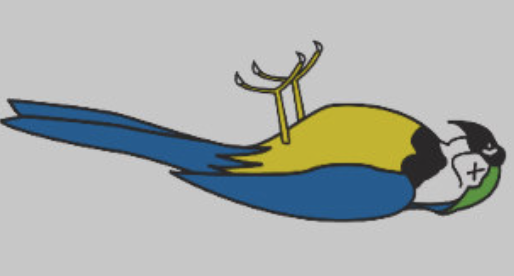
Vc = velocity of contaminant through the control volume

**BONUS POINTS: Using your MODFLOW model, develop a strategy (or strategies) which you could present to the Environment Agency for remediating the effect of the E-VIL contaminant – answer in word document as BONUS 2**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PART B – So you came through all that relatively unscathed? Congratulations. Do you want to kick back and sit on your laurels? or continue developing your skills and understanding?**

**At this point it is up to you to consider whether you want to extend your work. If you *might* be interested it would be good if you could at least indicate this as a possibility by emailing Fiona. Then have a look at the range of possible functions you could use to develop a more sophisticated model, perhaps of a real-life scenario, and we can discuss things further.**

Bored, cold and feeling nerdy?

Try running in a blank cell

import antigravity