GEO111

Week #06: Basic geochemical box modelling and reservoir dynamics

Time-stepping model of the Great Lake system

Instead of looking at the steady-state of the system, you might want to follow how the concentrations of heavy metals in the 5 lakes changes with time, perhaps due to a change in river flow, or a pollution incident. There is no analytical solution to the time-dependent response of the Great Lake system to a perturbation, so you are going to have to time-step through the simulated lake system, calculating the net change (i.e., loss or gain) of heavy metals for each of the lakes at each time-step. This will also illustrate something about what constitutes an 'equilibrium' situation or 'steady state' in a computer model – you never ever *quite* get there, so some 'judgement' is required as to where you draw the line and go off down the bar.

- First create a new MATLAB script.
- Create a loop structure; i.e.

```
for variable = expression,
    statements;
end
```

- We will assume that each time step represents 1 year. Start with a relatively short loop so that you can sensibly print stuff out to the screen (`disp`) to follow whether things are working and see what is going on. If you call the loop variable `t`, then a loop with `t = 1:10` will be sufficient to start with. Remember: start simple so that you can follow what is going on and debug it, and only later try and extend the (working) script to its final level of complexity.
- Before the start of the loop (but after you have commented what is does and cleared the workspace, of course – you weren't going to forget this were you … ?) create 5 variables to store the concentrations of heavy metals in the lakes and assign them all a value of zero. This is called *initializing* the variables. In some programming languages (such as FORTRAN) it can be very important to formally set variables you have created to zero before you start using them. You need something like;

```
cS = 0.0;
cM = 0.0;
…
```

- In case we need to change the input fluxes of metals to the lakes (maybe because of new data or a pollution reduction measure being enacted in one of the surrounding urban areas) we will use *parameters* to store the values of these fluxes. (Just like we defined a fixed reference parameter *b* in the

Excel exercise.) The values are given in the Table. Define the parameters (again, before the start of the loop) something like;

```
fS = 1000;
fM = 4500;
…
```

* You should comment this section suitably so that you can remember the flux units.

- Similarly, define the 5 parameters containing the river outflow rate from each lake;

```
rS = 63;
rM = 47;
…
```

- We will do one final piece of model initialization and also define the volumes of the lakes as parameters. Although the lake volumes might never change much, it is generally neater in computer code to have parameter names in the equations rather than numbers. And if you ever want to use the same number twice (or even more often than that), it is easier to remember the parameter name, then have to go look up the value to write down each time (particularly if it is a number with lots and lots of digits). The volumes of the lakes are given in the diagram. So, you'll want something like;

```
vS = 12221;
vM = 4871;
…
```

* Again, make sure that you comment the code appropriately so that you know what the units are.

- OK – now we are ready to start coding the main part of the numerical model. **To start with, we are just going to write the code to update the concentration of metals in Lake Superior (and only this lake).**

- The first thing to do is to add the heavy metal inputs to each of the lakes. Because the metal inputs are *fluxes* (i.e., the input is continuous), this addition must be done at each and every time step. (How the concentrations gradually decline over the following years following a one-off input (i.e., 'pollution') event is also a task for numerical simulation.) You should see that the code for making this metal input to the lakes will need to go inside of the loop structure. Yes? (Each successive year and each new time around the loop you will want to add a new bucket-load of metals pollution into the lakes.) OK, then we will proceed …

→ We know the concentration of metals in each of the lakes from the previous time-step `t`. We want to update the concentration for time-step `t+1`.

→ The new (time `t+1`) concentration of metals in the lakes will be equal to the previous concentration (time `t`) PLUS the *change* in concentration (Δconcentration) that reflects the addition of heavy metals from the surrounding urban areas. Δconcentration is equal to the amount you put in (Δmass) divided by the volume of the lake (it is as simple as; Δconcentration = Δmass / volume). Although Δmass is actually flux × Δtime, because the time step (Δtime) has length of 1 year, we can write; Δmass = flux × 1, or Δmass = flux.

→ For **Lake Superior**, take the concentration variable, add the increase in concentration (Δconcentration) to this variable, and assign this new value back to the concentration variable. The code for this should look something like;

```
cS = cS + fS/vS;
```

* Does this make sense? Ask if you are nor sure before going on.

- You'd better check that sensible things are happening before we add any more code. Print some stuff to the screen to check how things are progressing;

  → At the start of the loop write something like;

```
disp('year')
disp(t)
```

  * It would be nicer to have what the value is as well as the value itself all on one line. We can create a single string to display by converting the number value to a string, and then concatenating it with the text label;

```
disp(['year = ' num2str(t)])
```

  → Try running the script now. You should get a list of years from 1 through 10.

  → Now add some more useful display output to follow what is going on. At the start of the loop (just after the first `disp` command) add;

```
disp(['old concentration = ' num2str(cS)])
```

  and after the line updating the lake heavy metal concentrations, add;

```
disp(['new concentration = ' num2str(cS)])
```

  → Now run the script again.

  * You should see the concentration of metals in Lake Superior increasing year on year. Note that the 'old' concentration for any year `t+1` is the same as the 'new' concentration from the previous year (`t`).

  * When you are happy, comment out (`%`) the `disp` lines.

- Rather than printing stuff to screen it would be much nicer to see a graphic plot of what is going on. To plot how the metal concentrations evolve with time we will need to first store the information needed for the plot in an array. You can call this array anything you like. The array will initially contain just two columns of numbers – the first for the year, and the second for the concentration of heavy metals in Lake Superior. If the array is called `hello_kitty` (but please call it something more useful than this …) then at the end of the loop you would write;

```
hello_kitty(t,1) = t;
hello_kitty(t,2) = cS;
```

  All this is saying is that in the first position (column #1) of row `t` we assign the time-step number (which is equal to the year since the start of the model). In the second position (column #2) we will assign the concentration value.

  * Make sure that you understand why the '`t`' appears in '`hello_kitty(t,1)`' (i.e., how we are using the increasing value of the loop counter variable `t` to keep adding new rows to store the newly updated concentration values).

- Go run this. You should notice that an array '`hello_kitty`' appears in the MATLAB Workspace window. It should have size ('Value') of 10×2. Look at the array contents in the MATLAB Array Editor (or type `size(hello_kitty)`) and check that the concentration values are the same as you were previously printing out to the screen. Go debug if it if things are screwed up ☹

- At the end of the script plot the graph of time (year) against concentration. Make sure that everything is labelled etc as it should be. There are no prizes in life for being a lazy muppet.

- Be honest – is this not the most boring graph you have ever seen? Increase the number of time-steps the model runs through to 1000 (i.e., in the `for` loop definition). Re-run. OK; so it hasn't got any more interesting …

- What we haven't got yet is any way for the pollutants to leave the lakes. This is why the concentrations just rise linearly for ever and ever and ever. Losses to the lakes occur because the lakes drain out, either into other lakes, or (in the case of Lake Ontario) to the sea. We need to represent the losses of heavy metals from each of the lakes due to this drainage. Within the loop, where we update the lake heavy metal concentration we now want to take into account losses due to river drainage. The flux of metals out of each lake will be equal to the concentration of metals in the water times the water flow rate. Yes? So for Lake Superior, the loss of metals from each lake due to drainage will be equal to `rS×cS`.

  The units of loss are kg yr^{-1} just like the flux input (units are kg km^{-3} × km^3 yr^{-1}, which equates to kg yr^{-1}). We can therefore simply update our equation to take into account drainage loss as well as flux input as follows;

  ```
  cS = cS + fS/vS - rS×cS/vS;
  ```

  * If you wanted to, you could neaten up this equation by moving a factor of `('/vS')` outside of the net flux term `(fS - rS×cS)`.

- Re-run the script. Note that things are now much more interesting (ha!), and after 1000 time-steps (1000 years) the concentration of metals in Lake Superior is levelling off and not increasing very quickly. Try increasing the number of time-steps to 10000. Now you can clearly see that the concentration of heavy metals in Lake Superior is reaching what is know as *steady-state* (or it is in *equilibrium*), where the flux of metals in is exactly balanced by the flux of metals out and there is no longer any net change in the quantity (or concentration) of metals in the lake. You should also note that as the system is currently set up there seems to be little point in running the model for more than 1000 or 2000 time-steps, because very little change occurs after that. It is up to you to decide that any further change that occurs is trivial and can be ignored – perhaps when 95% of the final change has been achieved, or 98% or 99%. It will depend on the situation and how computationally expensive the model is to run. If this model took 10 hours to run 10000 years, you would certainly say that 1000 years was enough. Whereas it is actually so quick that running out to 2000 or even 10000 years is not a problem. Obviously another consideration is whether readers of your pollution report would be at all interested in a graph that had the interesting change bunched up at the left-hand end and nothing happening for most of the plot. Just because you can calculate 100000000s of years doesn't mean that you need present it to others. Some thought is required as to what information you want to get across in a presentation of a model simulation.

- **NOW**, add the (similar) code to update the metal concentration (due to flux input and drainage output) in the other 4 lakes. Also extend the code for storing the lake metal concentrations in columns 3 through 6 in the data storage array;

```
hello_kitty(t,1) = t;
hello_kitty(t,2) = cS;
hello_kitty(t,3) = cM;
```

…

* You will need to remember which column number corresponds to which lake. A comment in the code would be idea for this.

- We could plot all the data on the same graph to compare what is going on. Remember the '`hold on`' command in MATLAB that allows you to overlay other plots on the same figure? At the end of the script where you are currently plotting the evolution of metal concentration with time for just lake Superior, add additional lines for the other lakes;

```
plot(hello_kitty(:,1),hello_kitty(:,2),'k-')
hold on
plot(hello_kitty(:,1),hello_kitty(:,3),'r-')
plot(hello_kitty(:,1),hello_kitty(:,4),'g-')
plot(hello_kitty(:,1),hello_kitty(:,5),'b-')
plot(hello_kitty(:,1),hello_kitty(:,6),'c-')
```

* The code to get a continuous black line for Lake Superior can be passed as a parameter to the `plot` function – it is '`k-`' (see MATLAB help on 'LineSpec'). The 2nd, 3rd, 4th, and 5th lines are colour-coded – red: Michigan; green: Huron; blue: Eries; and cyan: Ontario.

- Go run the script again. Notice how the different lakes have different *response times* (i.e., the time taken to reach *equilibrium*).

- This Lab does go on and on and on … but very *very* almost there. There is just one final thing missing. When water flows into the lakes Huron, Erie, and Ontario, it is carrying with it heavy metal pollution from the lakes upstream. We need this last component in the model. For these three lakes we then need to introduce an additional flux. Fortunately, we have already calculated this flux – it is equal to the loss rate from the lake upstream. For example, the additional metal flux via river flow to Lake Ontario is equal to the river loss flux out of Lake Erie. Look at the diagram of the lake system to see how the lakes are connected. For Lake Ontario, for instance, we need to make one final modification to the mass balance calculation in the loop;

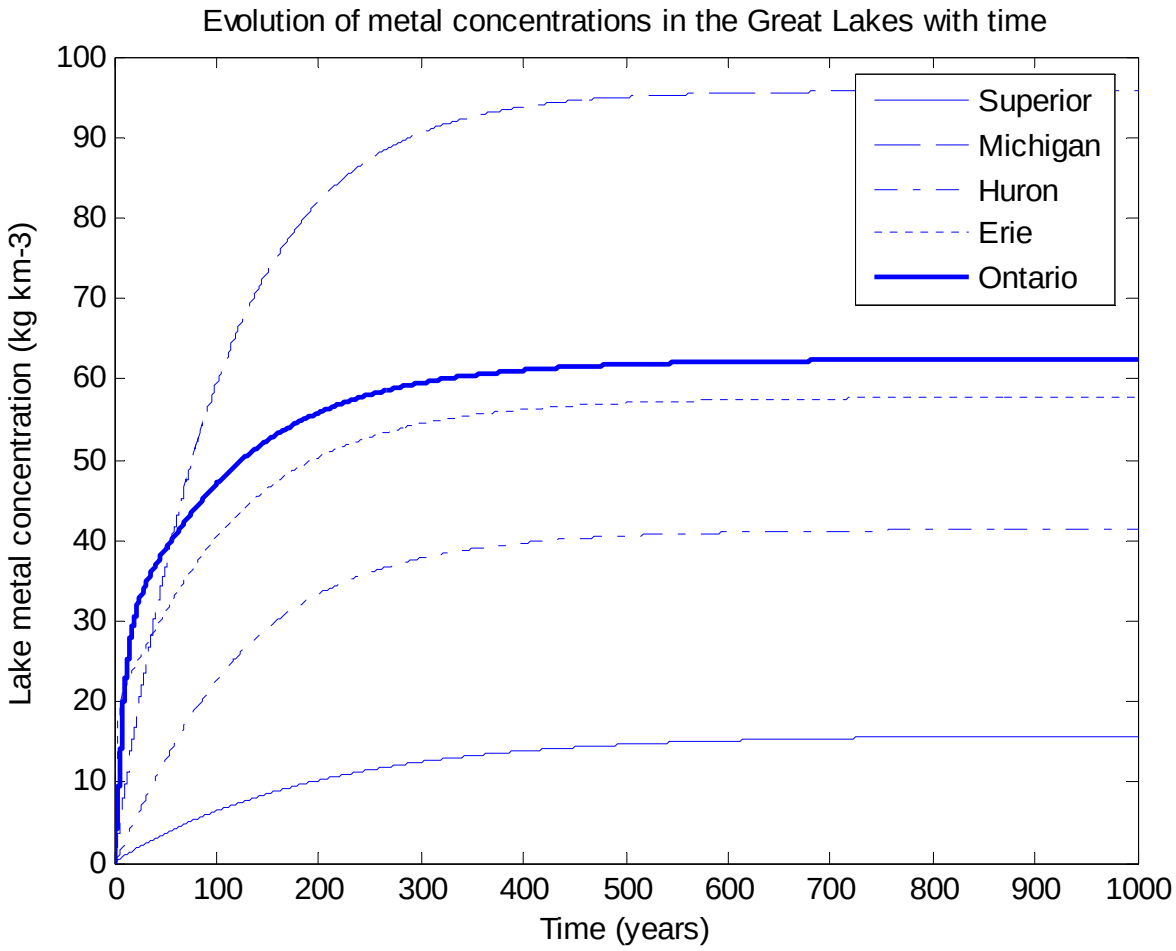
```
c0 = c0 + f0/v0 - r0×c0/v0 + rE×cE/v0;
```

* All this say is that; the concentration of metals in Lake Ontario next year, will be equal to the concentration this year (cO) PLUS the metal flux input (`f0/v0`) MINUS the drainage loss (`r0×c0/v0`) PLUS the river input from lake Erie (`rE×cE/v0`).
* Again, you could shorten the equation if you want to by taking out common factors.

- Also write the code to complete the mass balance for Lake Erie (gain from Lake Huron) and to Lake Huron (gains from Lake Michigan AND Lake Superior).

- Re-run your script. Hopefully you'll see something like this below (if I haven't screwed up). Note that in a black-and-white such as this you can see how setting different line styles to the lines (for example; dot-dot, dot-dash) (or using point markers if you had fewer points in the lines) can be really useful. See MATLAB help on 'LineSpec' for more. Maybe try some of these out. See help on 'Plot' on how to pass the parameters you want.

- Check that the final 'steady-state' (or close to steady state) metal concentration values are close to your analytical solution from previously.


Evolution of metal concentrations in the Great Lakes with time
Lake metal concentration (kg km-3)
Time (years)
Superior
Michigan
Huron
Erie
Ontario
0
10
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800
900
1000