mfLab how-to's

Theo Olsthoorn

4 January 2012

Contents

1 Introduction

mfLab is a strong modeling concept and environment by combining the strengths of Matlab and the standard and robust finite difference groundwater flow and transport models MODFLOW, MT3DMS SEAWAT and the wealth of packages made for them. However, to become a skilled modeler, most of us need examples and be shown how to do things in this environment efficiently. This may prevent a lot of frustration.

mfLab models are typically made by copying an existing example and adapting it to one's needs. Therefore, we need examples of a wide spectrum of usage. Many examples can be found in the examples directory that comes with *mfLab*. These examples are generally well documented with interspersed comments. However, all too many comments also distracts from the essence and makes m-files needlessly long. Therefore, some basic skills in mfLab modeling or modeling in Matlab in general, can best be taught from a separate how-to manual, that this one tries to present.

The subjects have not been made nor ordered in a systematic way. Rather they have been made while modeling myself. This manual will therefore, be extended regularly when new approach are made and examples generated.

2 Making a grid

A grid can be made by hand, by typing in numbers, or by using available functions, or by a combination. *mfLab* requires an `xGr`, `yGr` and a `zGr` vector. `zGr` may also be a complete 3D array. In that case, `zGr` specifies the top and bottom of each and every cell in the model.

Having specified the grids, there are two functions to make sure that `xGr`, `yGr` and `zGr` are sorted, oriented in the right vector direction and that duplicates are removed.

```
[xGr,yGr,xm,ym,Dx,Dy,Nx,Ny]=modelsize(xGr,yGr);
```

or

`[xGr,yGr,zGr,xm,ym,zm,Dx,Dy,Dz,Nx,Ny,Nz]=modelsize3(xGr,yGr, zGr);`
the m in `xm`, `ym`, and `zm` indicates the vectors of the cell centers, the `D` in `Dx`, `Dy` and `Dz` indicate the vectors of cell sizes and the `N` in `Nx`, `Ny` and `Nz` indicate the number of cells in the corresponding grid directions.

`yGr` and `ym` vectors will be oriented high to low, so that the first line in the arrays correspond with the most northerly position. Likewise, `zGr` and `zm` are oriented from high to low to make sure that the first line (or first plane) of an array is the layer with the highest elevation.

You may plot your grid as follow

```
figure; hold on;
plotgrid(xGr,yGr)
```

To facilitate making a grid, rather than typing numbers you may use Matlab's `linspace` and `logspace` functions. See their documentation.

mfLab comes with the function `sinespace` to add details and generate smooth transition between parts of the grid around objects (see directory mflab/mfiles/gridcoords) and type `help sinespace`.

`[x,dx]=sinespace(x1,x2,N,alfa1 [,alfa2])`

divides the axis between and including x1 and x2 into N+1 sections, N gridlines, with section lengths according to the the sine function. If alfa2 is left, out it is interpreted as `alfa1=0` and `alfa2=alfa1`. So to refine the grid towards `x2`

`[x,dx]=sinespace(x1,x2,N,pi/2,pi);` % refines the grid towards `x2`
`[x,dx]=sinespace(x1,x2,N,0,pi/2);` % refines the grid towards `x1`
`[x,dx]=sinespace(x1,x2,N,0,pi);` % refines the grid towards `x1` and `x2`, coarse in the middle

If grid coordinates are generates in arbitrary ways, involving many a fine grid around wells for instance inside a courser grid, that itself also honours the details of a local stream end so on, then, one all these coordinates are put together, one may expect a very irregular grid, not only with duplicates but, especially also with near duplicates. Such small cells are rather merged with larger neighboring cells to make sure no cells are smaller than some specified minimum cell size. The function cleangrid can do the job (be it in a bit simplistic way). Especially the computation time of transport models strongly depends on the minimum cell size.

`xGr=cleangrid(xGr,dxmin);`

The can be repeated for the yGr and zGr directions in the same way, using dyzim and a dzmin.

3 Parameters in the workspace

Some parameters must and many parameters may be specified in the Matlab's workspace by mf_adapt, so that mf_setup finds and uses them to generate model input. Such paramters are CAPITALIZED and have the name of the same parameter in the MODFLOW, MT3DMS and SEAWAT manuuals. For instance, HK is used by the LPF package (horizontal conductivity). Thefore, if the LPF package is set to "on" in the NAM worksheet of the workbook pertaining to this particular model, then mf_setup expects to find it as a 3D array in the workspace. If it is not there, you will get an error and *mf_ setup* will halt.

However, if you use the BCF package instead of the LPF package, than mf_setup expects to find the array TRAN instead of HK in the workspace for the transmissivity of the layers and HY for the conductivity of the first layer if it has a free water table. And so on. What each package requires can thus be taken from the original MODFLOW, MT3DMS and Seawat manuals.

Some parameters you may also specify in the worksheet LAY. That means that there will be only one value for an entire layer. Some parameters only have one value per layer, such as LAYCON„ but others may have NROW times NCOL values per layer, while it may be convenient most of the time to just use one. For instance the CHANI, horizontal anisistropy is an example on the LAY sheet and RECH, recharge, EVTR, are examples on the PER sheet. mfLab naturally extends the functionality of adjoint parameters in the worksheet. For instance, INRECH defines whether or not RECH should be read in a given stress period. If INRECH<0 recharge rates form the preceeding period is used and if INRECH>0 an NROW by NCOL array of recharge values will be read. mfLab addes flexibility by redfining the meaning of INRECH=0. In that case, that is if for a certain stress period INRECH=0, then mf_setup will take recharge for that stress period from the workspace instedad of using the single value in the worksheet for the entire model. This type of interpretation is also true for many other parameters that have a single value per stress period in the PER workhsheet en for some in the LAY worksheet.

To provide maximum flexibility and prevent confusion in cases where an arbitrary mix of values per layer from the worksheet and for all cells in the workspace will be used, some rules are required. They are as follows:

If the values for any of the stress periods or layers are to be obtained from the workspace, than there must a a value for all layers and spreadsheet. This way there can be now confusion about which layer or stress periods will be taken. However, since only specific layers and stress periods will be taken from the workspace and the others form the worksheets, the values specified for the layers that will not be used are immaterial. So you may generate a recharge array in the workspace, which must be called RECH (because this is the name of the parameter used by MODFLOW), and it will have NROW by NCOL by NPER values. NPER may be replaced by the highest stress period number that wants values from the workspace. The values in RECH for any stress period not read from the workspace may be anything, such as NaN. Similarly for layer values to be partially read from the workspace, mf_setup expects to find NROW by NCOL by NLAY values, in which NLAY may be replaced by the highest layer number of which data will be read from the workspace instead of worksheet LAY. Values in this array for any layer that will not be sought in the workspace are, again, immaterial.

To add further flexibility and to reduce the space required to hold possibly large arrays line NCOL by NROW by NPER in the workspace, there it is also allowed to specify the values in a cell array, with one cell per stress period or per layer. These cells may be empty, except for those layers and stress periods for which values will be taken from the workspace. It is even allowed to have a single value in cells corresponding to layers and stress periods that are required in the workspace. In that case, these vlaues will be interpreted as valid for an entire layer. This flexibility reduces redundancy to the maximum possible and yet prevents confusion about which layers and or stress periods to be read.

The parameters that will be recognized as model parameters by mf_setup can be found in the top of that script. For some parameters there are alternative names, which maintains backward compatibility and adds name flexibility. This way DELR and DX are equivalent, for example.

Boundary conditions like WEL, DRN, GHB, RIV may also be defined both in their worksheet and in the workspace. However, the more experience is gained the less defining them in the worksheet is a good idea. The reason, it contrasts with the basic advantag of mfLab, its grid indepedence and flexibility to parameterize as much as possible. It is generally much easier to defined them directly in the workspace and for that use some of the functions provide to get the required cell indices for wells that are given in real-world coordinates. Look at some of the examples. The respective worksheets are left in the workbook also as a convenient reference of the format that the input requires (which column contains what).

These boundary conditions can be specified in two ways.

1) As a list of values where the first column is the stress period, so that mfLab can select those belonging to a certain stress period and you can mix the input to your liking. The other columns 2:end are the same as described in the MOFLOW manual. This way the parameter WEL in the workspace is an array with one line per well cell and the following columns:

LAY ROW COL Q

Therefore this becomes

IPER LAY ROW COL Q

in *mfLab*. If there are stress peirods without wells, mfLab will see that because the corresponding IPER values will be missing in the array. Where values of a previous period are to be used, and MODFLOW inserts just -1 in the input file, use

–IPER LAY ROW COL Q

where LAY ROL COL Q may be anyting, even NaN. mfLab uses the negative IPER value to signal that period IPER will use all values of period IPER-1. However Matlab requires that all lines in the array have the same numer of columns.

2) The second method is to define it as a struct array, with fieldname “values” where each element of the struct will contain all values for that stat stress period, like

WEL{IPER}.values

It is clear that now the values have the from

LAY ROW COL Q

without the stress period number, which is already impiled in the element number of the well struct.

Stress periods without values will have their values field empty.

Stress periods using the previous stress periods’s values may use negative values for their layer. The numbers themselves are then immaterial.

4 Running scripts off line

When running scripts like mf_lab, mf_analyze and, especially mf_setup, matlab waits until the job is done, meanwhile receiving output form Modflow, or Seawat running in the background. Waiting for jobs to finish, may be quite frustrating as you're out of control for time time. I may thus be useful to run the job in the background while you continue working on something else in the mean time.

If you have access to the parallel computation toolbox, you may run one or more jobs in parallel in the background using

```
job=batch('mf_setup','Pathdependencies',mf_mybatchpaths
```

The parameter value pair arguments are necessary because otherwise mf_setup running in its separate environment cannot find the mfiles of mfLab. mf_mybatchpaths is just an m-file in the mfiles/etc directory, which contains the paths to the mfiles as in the shortcut

```
function paths=mf_mybatchpaths
% MF_MYBATCHPATHS Sets paths when running script off line using batch()
%
% USAGE mf_mybatchpaths
%      job=batch('mf_adapt','PathDependencies',mf_mybachpaths);
%      wait(job); % to see when it ends, but then you loos control of your PC, just skip th
%      job        % to see if it is ready.
%      load(job); % when ready to load all parameters back from the
%                   off-line workspace into the original workspace
% When running a script in batch node, off line, it will run in its own
% workspace environment and does not know about paths set in the original
% one. Therefore we have to set the paths to the mfiles director of mflab
% and pass these path names onto the batch job's environment
%
% Notice that batch uses the parallel toolbox
%
% TO 110511

P='/Domain/tudelft.net/Users/tolsthoorn/GRWMODELS/mflab/';
paths=...
   {[P 'mfiles/read'];...
    [P 'mfiles/write'];...
    [P 'mfiles/etc'];...
    [P 'mfiles/visualization'];...
    [P 'mfiles/gridcoords'];...
    [P 'mfiles/fdm'];...
    [P 'mfiles/analytic'];...
    [P 'bin']};
```

5 Some useful parameters in mf_adapt: AXIAL, BACKGROUND, GREP, AFTERMFSETUP

By setting the parameter AXIAL, this is done by including the line

```
AXIAL=1;
```

in mf_adapt, mfLab will compute the model axial symmetrically. Note that the coordinates are multiplied by $2\pi|xm|$ so that x=0 is the center of the model. This is useful for models having only one row in y-direction.

By setting the parameter

```
BACKGROUND=1;
```

in mf_adapt, the models will execute in the background. This means you can immediately continue working in Matlab without having to wait for the model to finish. The consequence though is, that you will not see screen output from the model in your command window, and you must figure out yourself when the model

has finished. Normally this is the case when the fan of your computes has become quite again. Check the tail of the list file or check if the process is still running (list of processes in Windows can be seen when pressing CTRL-ATL-DEL or by the system call

```
system('ps');
system('ps | grep swt');
```

on the Mac or on Unix.

Set parameter

```
GREP='filter string'
```

in *mf_adapt* to filter the command window output of SEAWAT by the 'filter string'. 'filter string' is any string that can the output contains. 'PERIOD' is a useful string here to follow conveniently how SEAWAT progresses.

Lastly you may set the parameter AFTERMFSETUP with a string as argument which is the matlab instruction to run of ter mf_setup. Normally you would not set this parameters, but first check whether the model has normally terminated. If you are sure that will happen in your case, you could include this parameter in mf_adapt to immediately run mf_analyze when mf_setup is ready:

```
AFTERMFSETUP='mf_analyze';
```

So three lines of your mf_adapt may look like this

```
AXIAL=1;              % interpret input as axially symmetric
BACKGROUND=1;         % execute in model in the background
GREP='FROM TIME';     % only show output lines containing 'FROM TIME'
AFTERMFSETUP='mf_analyze'; % immediately follow
                           % up with command after mf_setup
```

I'm not in favor of using AFTERMFSETUP because of the model fails for any reason, you may not see it and apply mf_analyze on the old or possibly wrong data. But it is sometimes useful in production work and automation.

6 MULTIDIFFUSION

Diffusion coefficients are specified in the LAY sheet of the workbook under the heading DMCOEF, hence per layer. However, MT3DMS allows DMCOEF to be specified on a cell-by-cell basis. Therefore, you can specified DMCOEF as a parameter in *mfLab* workspace as well as in the LAY worksheet. The parameter DMCOEF must be a cell array of length NLAY that has a matrix NROW,NCOL of diffusion coefficient values for each layer that needs to be specified. Which layers to specify in the workspace is deduced from the column DMCOEF in the LAY sheet. If the value of a layer in the worksheet is $>= 0$, then the value in the worksheet i used. If, however this value is < 0, then the value in the cell array is used, taking the cell that corresponds to the layer being processed. If DMCOEF in a layer in the worksheet < 0, mfLab requires the matrix DMCOEF to be present in the workspace. Clearly, layers that are specified in the worksheet LAY may correspond to empty cells in the cell array DMCOEF.

For convenience of the user, mfLab also accepts a regular array of diffusion coefficients of size (NROW, NCOL, NLAY, NCOMP) where NCOMP the number of species each with its own diffusion coefficient.

7 Managing multiple colormaps in the same axis

This is a subject that is discussed often. For instance one desires a figure with both temperature and chloride data which have completely different data-value ranges. Because the Malab's colormap is a child of the figure, colors in the existing axies will change if new data is plotted having data values outside those of hte data already in the figure. This is because the CLim property of the figure will be extended to accomodate the new data, een if they are plotted on a different axis on the same figure. This also happens often, if one want to plot surface contours of for instance temperature on an image or photo showing the background, even if the two are on different axes, but within the same figure. How this can be overcome is explained in Matlab's help documentation, but it seems still confusing at times.

These problems only concern data objects whose colors are plotted using the figure's colormap. The data then consist of a 2D-array with indices into this colormap, ether directly or scaled. With direct indexing, each

value in the data array is an index into the colormap, i.e. the line of the three column RGB colormap array. With scaled indexing, each value in the data array is first scaled according to the clim of the axis (which is an axis property), that is. the line in the color map is found by interpolation of the data value between the clim values set for that axis. Whether or not scaled or direct indexing is used depends on the CDataMapping property of the image (not an axis property). This is set by

```
image(X,'CDataMapping','scaled') or direct
```

or

```
imagesc(X,'cLimits',[CLim1 CLim2]);
```

The latter allows setting the clim values at the same time, whereas the first requires setting them through

```
caxis([cLim1 cLim2]);
```

or

```
set(ax,'clim',[cLim1 cLim2]);
```

The latter three comments also set the CLimMode to manual, so that it does not change with addtional data plotted on the same axis.

By setting the values of cLim1 and cLim2 for the current axis, one can determine which portion of the colormap is going to be used by the data. By doing this for different axis indidually, one can make sure that each axis uses a dedicated portion of the overall colormap of the figure, and, therefore, its own color set. To combine different colorsets in place, one overlays the axes precisely. This way arbitrary-valued data sets can be combined in the same figure and apparently on the same axis.

Before elaborating the procedure to compute the correct climits in an axis with multiple colorsets on the same figure, we have to mention that it is also possible to plot colors directly without use of a colormap. The data to be plotted then have to consist of a NxMx3 arrays containing the R G and B value of each 2D pixel. Direct use of RGB is always applied when specifying color [R G B] or the predifined shortcuts 'b' r' 'g' 'k' 'm' 'c' 'y' 'w'. So if we wish to plot surface contours on top of an image, we may considere plotting the image using RGB directly on one axis (or even the same axis) and use indexed plotting for the surface contours.

What is explained here can be seen at work in the example "Mijdrecht" under examples/swt_v4/. The example animates saltwater movement in a cross section below polder Mijdrecht, The Netherlands, which was put dry around 1870 and ever since discharges substantial amounts of brackish to saline groundwater. Parameters used are only approximate. To run the exmaple, which simulates 150 years development in yearly steps takes about 8 minutes, depending on the speed of your computer.

Matlab can only use one colormap per figure, even if this figure contains multiple axes Plotted in an axis are mapped over this colormap according to the clim property of the axis. By default this property is set automatically depending on the total range of plotted objects like surfaces, filled contours and images. As a consequence adding additional objects to a figure may change the colors of already plotted objects. It may thus be difficult to plot objects to their desired colors, as typically each object or axis would like to use its own colormap.

The solution is to concatenate colormaps and to make sure that specific objects use the correct portions of the thus obtained overall colormaps. As each index in a colormap may have an arbitrary color defined by its RGB values, everything is thus possible.

Matlab's help documentation addresses this under "Simulating multiple color maps in a figure", where examples are given.

Figure 1 shows the total index range of the figure's color map, $i_1 = 1$ to i_4. We may have constructed this colormap ourselves by contatenating several other ones. Now assume that we want to use the portion from index i_2 to i_3 of this colormap, and that this portion maps to the data values c_2 to c_3. The only thing we have to do is to set the clim of the axis to c_1 and c_4. Therefore, we just have to compute c_1 and c_4 given the know size of the overall colormap, the desired portion to use and the disired data values range to use of a the axis. Remember, the colormap corresponds to the figure, but the clim is set of each axis on the figure separately.

To match a colormap index range $i_2 \ldots i_3$ to the data range $c_2 \ldots c_3$ we may setup the following linear relationships between the two (see figure 1)

$$\begin{aligned} c_2 &= c_1 + \frac{i_2 - i_1}{i_4 - i_1}(c_4 - c_1) \\ c_3 &= c_1 + \frac{i_3 - i_1}{i_4 - i_1}(c_4 - c_1) \end{aligned}$$

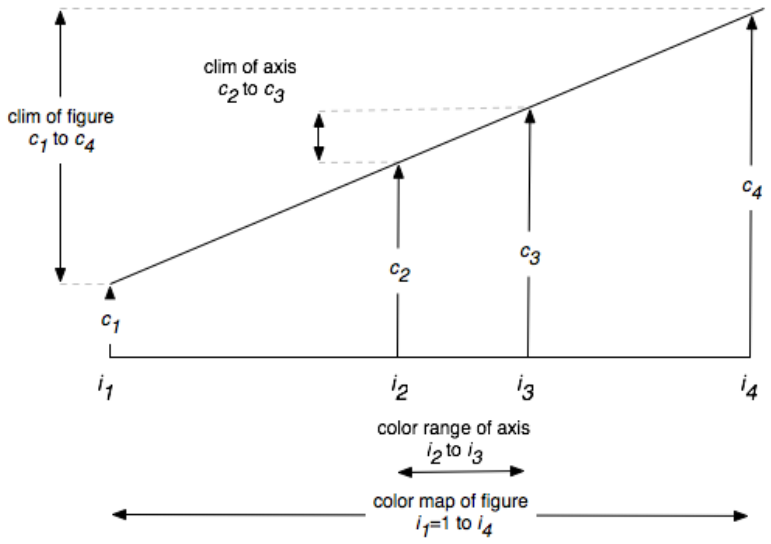

Figure 1: Data values c mapped to color map indices i. We have data values c_2 to c_3 that we wish to color using a portion with indices i_2 to i_3 of the total color map with indices i_2 to i_3. This we will achieve if we set the value limits of this axis to c_1 and c_4.

From which c_1 and c_4 can be solved

$$\begin{pmatrix} c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \left(1 - \frac{i_2 - i_1}{i_4 - i_1}\right) & \frac{i_2 - i_1}{i_4 - i_1} \\ \left(1 - \frac{i_3 - i_1}{i_4 - i_1}\right) & \frac{i_3 - i_1}{i_4 - i_1} \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_4 \end{pmatrix}$$

or

$$\begin{pmatrix} c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \frac{i_4 - i_2}{i_4 - i_1} & \frac{i_2 - i_1}{i_4 - i_1} \\ \frac{i_4 - i_3}{i_4 - i_1} & \frac{i_3 - i_1}{i_4 - i_1} \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_4 \end{pmatrix}$$

$$\begin{pmatrix} c_2 \\ c_3 \end{pmatrix} = \frac{1}{i_4 - i_1} \begin{pmatrix} i_4 - i_2 & i_2 - i_1 \\ i_4 - i_3 & i_3 - i_1 \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_4 \end{pmatrix}$$

and with Matlab's backslash operator, we have the values c_1 and c_4 to set the clim of the axis

$$\begin{pmatrix} c_1 \\ c_4 \end{pmatrix} = \frac{1}{i_4 - i_1} \begin{pmatrix} i_4 - i_2 & i_2 - i_1 \\ i_4 - i_3 & i_3 - i_1 \end{pmatrix} \backslash \begin{pmatrix} c_2 \\ c_3 \end{pmatrix}$$

so

$$set\,(ax,\ \text{'clim'},\ [c_1,\ c_4])\,;$$

will do the work. *mfLab* has the function *mf_ clim* to compute the clim values c_1 and c_4.

7.1 Example

First construct the desired colormap form a set of maps

cmap = [cmap1; cmap2; cmap3; ... cmapN];

to be used in axis ax_1 to ax_N. the indices i_2 and i_3 and i_4 are now known wihle always $i_1 = 1$. The data values for each axis are also known, because they are user-specified. Then, setting each axis to map to the correct colors can be done as follow

```
colormap([cmap1; cmap2; ... cmapN]);
L=size(colormap,1);
for ia=1:length(ax)
        set(ax(ia),'Clim',mf_clim(c2(ia),c3(ia),i2(ia),i3(ia),L));
end
```

7.2 Constructing suitable colormaps

The easiest way to construct suitable and pleasing colormaps is by direct use of the ones prepared by matlab. See help documentation under Supported colormaps. For use with density modeling, showing high concentrations of salt as read and low ones as blue in indicate pure water, the colurmap jet is probably always useful. A colormap may be created of any length by specifying the length as the argument of the map. To create a 32 index long "jet" or default colormap use this:

```
colromap('jet(32)'); % with quotes
colormap(jet(32));    % without quotes
colormap([jet(24);hot(32);winter(64)]); % concatention of color maps
jet(44) % produces the jet or default color map
colormap % produces the current color map (figure property)
```

Other supporte colormaps are

jet, hsv, hot, colorcube, flag, cool, spring, summer, autumn, winter, gray, bone, copper, pink, prism, white and lines

For instance one may one to use the jet colormaps to show concentration and use the gray colormap to show the conductivities of the layers.

```
c1(1)=min(crange); c2(1)=max(crange);
c1(2)=min(hrange); c2(2)=max(hrange);

colormap([jet(64); gray(32)]); I2=[1 65]; I3=[64 96]; L=64+32;

set(ax1,'Clim',mf_clim(c2(1),c3(1),I2(1),clim(I3(1),L));
set(ax2,'Clim',mf_clim(c2(2),c3(2),I2(2),clim(I3(2),L));
```

7.3 Using specific colors

Objects can be given specific colors like patches and lines by using one of the familiar color codes 'b', 'r', 'g', 'k', 'm', 'c', y', 'w' or a color defined by RGB values like

```
set(obj,'color',[0.5 0.2 04]);
```

where 'obj' is the handle of the object that can be obtaind as the output of the concerned funcion when called.

7.4 Using transparency

Objects can be made transparent by setting the property 'alpha' to a value less than 1 and larger than zero.

In the case of filled contours, we may have to set the 'alpha' values of the children of this object, which are patch objects that have this property. This may be done as follows

```
set(get(obj,'children'),'facealpha',0.5);
```

7.5 mutiple colormaps made easy in mfLab

mfLab provides some fucntions to make managing multiple colormaps relatively straightforward.

You first generate axes to plot you different data types on for wich you want a specific set of colorts to be used through their color maps. All we need for that is the axes handles, the data range within each axes and the colormap belonging to each axis. This put in a struct

```
clrstr(i).ax=ax(i);
clrstr(i).range=mf_range(arange);
clrstr(i).map=jeg2;
```

This is repeated for every axis
Then pass the clrstr to

```
mf_setmulticolormap(clrstr)
```

and your are set.
Now to plot a certain data type, first switch to the axis you want to use and plot your data
axes(ax(3);

```
[c,h]=fcontour(x,y,yourdata,isovalues);
set(h,'children','edgecolor','none');
% if you do this with contour instead of
% fcontcour you won't see your lines (why?)
```

end
For images use clrstr(i).range=[0 255].

8 Linking objects

This is useful when you want to rotate or zoom multiple axis or objext.
Create a link object with the handles involved in a single array, h

```
hlink=set(h,{'xLim','yLim,'zLim','view'});
```

Then if you change or zoom one of the involved objects, including axes you change them all.

```
set(h(2).'xLim'=[0  30]);
view(h(2),3);
```

9 Plotting your data on all sides of box with a cut-out

Visulization of complex volumetric data can be done in many ways one of the convenient methods is by plotting the colors on all sides of a box with a cutout to see some aspects of its interior.

h=mf_3Dblock(XM,YM,ZM,C,ix,iy,iz[,facealpha])

does just that. You have to provide full 3D arrayas of the coordinates and the value to show (C) and the cutout location in grid coordinates ix, iy, iz. This will plot three boxes and color all their sides so you can turn it as a 3D object. The boxes are in the following coordinates.

if ix, iy and iz are postive, then the x=coordinate of the first box is limited to 1:ix, the y-coordiante of the second box to 1:iy and z-coordinate of the the third to 1:iz. If genative values are used, the 3 boxes will run from ix:Nx, iy:Ny and iz:Nz respectively.

This allows to make any incision.

Notices, hat ix>1 and ix<Nx, iz>1 and iy<Ny and iz>1 and iz<Nz.

The last argument is optional, and allows to set the transparency. 0 is fully tranparent and 1 is opaque.

see

```
mfLab/examples/swt_v5/FFSE/FSSE_FHB for its usage.
```

10 Working with Google Maps and Google Earth

10.1 Introduction

Working with Google Earth and Google Maps give great opportunties for modelers. It is now possible to obtain a figure directly from google into Matlab, as a file or as a Matlab figure. We can therefore plot results directly onto a map, from which we can also measure coordinates. Presenting results on Google Maps (figure in Matlab)

is a great manner to convey more easily the results of our models. *MfLab* provides functions and examples that show how to do this.

I include the theory regarding the way to deal with the coordinates, as it was necessary to write the functions. The functions can be found under mflab/mfile/visualization. I deemed the theory necessary as good explanations are hard to find on the internet. Alstho the theory may be a bit tough, the testURL examples at the end make good for it. They show how easy it is to use Google Maps figures in Matlab and, therefore, in *mfLab*.

Dealing with Google Maps is worth a book. It seems nasty at times, but in the end it is phantastic. To learn more look in the documentation for use of static maps for details:

`http://code.google.com/apis/maps/documentation/staticmaps`http://code.google.com/apis/maps/documentation/

mfLab has several functions to deal with such maps. For example, it is easy to obtain an arbitrary figure of any location in the wold directly in matlab as an image, even a georeferenced one. However, we have to know a bit ofGoogle coordinates. Working in Lat Lon coordinates is well known. But on top of that GM works with tiles and within tiles with pixels measured 0:255 from the upper left corner to the lower right corner.. The detail and resolution of each figure depend on the zoom level. Zoom level 0 is the basic level, which encompasses the entire globe in a single tile in Marcator conformal projection. At the first zoom level we have 4 tiles, in the second 16 and so on.

You can use an *mfLab* function to get a figure from google earth directly into your figure or file

```
url.center=[-33.856857,151.215192];              % Sydney
url.zoom=16;              % must be between 0 and 21
url.size=[512  512];      % must be < 640
url.maptype='satellite';
URL=mf_GM2PIC(urlstruct,'satellite','png')
```

the last two arguments default to 'sattelite' and 'png' respectively. See the help of the funcion for details and how to fill in the url-struct.

The current version of mf_GM2PIC allows features line marker and paths and fills (paths with fillcolor set).

With an output argument, nothing will happen, only the URL will be printed. This allows you to check it. Without an argument, the a figure is requested and put in the webbrowser.

To get it as data direcly, use

```
A    = imread(URL); % get it directly into matla
info = imfinfo(URL); % get associated image info
image(A); colormap(info.Colormap);
```

The figure will be centered around the center you provide. However, this center is generally not the same as the zero of the tile form which google computes its coordinates.

The function

```
[xPix,yPix,ix,iy]=mf_GMLLtopix(LAT,LON,zoomlevel)
[xPix,yPix,ix,iy]=mf_GMLL2pix(LAT,LON,zoomlevel,)
```

This gives the pixel coordinates relative to the UL corner of the tile that contains this LAT LON poitn at the given zoom level.

You may find the center of that tile with

```
[LAT LON x y s]=mf_GMpix2LL(ix,iy,zoomlevel,xpix,ypix)
```

by setting *xpix* and *ypix* both to zero and using the same zoomlevel and the ix and iy obtained by the preveous call.

The other output arguments of this function are the coordinates in m of the input relative to the UL corner of the metioned tile. This allows to get coordinates of any picture in m readily.see

test_GM2PIC_3.m

in the directory

mflab/mfles/visualization

for an example of the use fo thse functions.

There is a bunch of mfiles in the visualization directory, some of which will be used below. They all deal with Google coordinates one way or the other.

10.2 How to get Google Earth coordineates into Matlab?

To get GM coordinates into Matlab, you can make a path in GE (Google Earth) and after finishting it, exort it through save as an

```
yourpath.kml
```

file in the directory of you liking.

When finsished get the WRS84 coordinates of Google Maps with

```
[E,N]= mf_kmlfile('yourpath.kml');
```

Notice the Easting Northing if reversed compared to Lat Lon.

Tou can transfer these coordinates into the Dutch system by

```
[XNL,YNL]=wgs2rd(E,N);
```

You may also directly tranfer the path to RD coordinates

```
[XNL,YNL]=mf_kmlpath2RD('yourpath.kml');
```

To tranfer Dutch RD coordinates to WGS use

```
[E,N]=rd2wgs(XNL,YNL);
```

10.3 Google Maps coordinates

Google Maps usees Mercator projection to put the earth on the screen. This projection casts points at the earth's surface along lines through the earth's center onto a cylinder fitting the earth at the equator having its axis through the poles. This means that the poles map at infinity, which requries some practical limit for the maximum and minimum latitude to address, which we will give below. The projection lines through thte center of the earth and the points on the surface are will undergo a further transformation to make the mercator projection is that it is conformal. This means, that the within infinitesimal small areas, the ratio of the length of line pieces in the original and projectd space is independent of their direction and further that the rotation of such lines by the projection is also independent of their directions. This ways, straight lines on the globe in a given direction remain so on the while locally mapped objects maintain their shape.

As in all map projection math, longitude and latitude are transfrerred from degrees to radians first $\lambda = \frac{\pi}{180} LON$ and latitude or northing to $\phi = \frac{\pi}{180} LAT$

Mathematically, the requirement that a mapping is conformal may be expressed by the constant stretching σ and rotation α in the local neighborhood of any considered point. That is, the stretching and rotation may and will vary over the map, but locally is is constant, so that all directions are rotated over the same angle and all distances are stretched by the same factor. This can be expressed as follows

$$\begin{pmatrix} dY \\ dX \end{pmatrix} = \sigma \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} d\lambda \\ d\phi \end{pmatrix}$$

Where X, Y denotes a projected point on the cylinder (but with radius 1, i.e. divided by R) and λ, ϕ a point on the globe.

Choosing$\alpha = 0$, gives

$$\begin{pmatrix} dX \\ dY \end{pmatrix} = \sigma \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d\lambda \\ d\phi \end{pmatrix}$$

Given that dx and dy stretch differently in the projection, the basic projection along straight lines does not comply with this expresson. To turn it into a conformal map, we have to tranform one of the axes separately, which, obviouwly will be the y-axis or the ϕ-axis for that matter. Therfore wrting $\sigma = \begin{pmatrix} 1 \\ \sigma \end{pmatrix}$ to stretch the ϕ axis separatel in advance, we get

$$\begin{pmatrix} dX \\ dyY \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d\lambda \\ \sigma d\phi \end{pmatrix}$$

If σ is constant we have a conformal map. So we now have

$$dY = \sigma d\phi$$

while, from the straight-line Mercator projection we have

$$dY = \frac{d\phi}{\cos(\phi)}$$

Clearly, dY depends on ϕ, but locally ϕ can be considered constant and therefore this transformation makes the mapping conformal.

Integratin , with as boundary condition $Y = 0$ at $\phi = 0$, yields the y-axis transformation for the conformal Mercator projection:

$$Y = \ln\left(\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)\right) \quad (1)$$

$$Y = \frac{1}{2}\ln\left(\frac{1+\sin(\phi)}{1-\sin(\phi)}\right) \quad (2)$$

in which $-\infty < Y < \infty$ for $-\frac{\pi}{4} \leq \phi \leq \frac{\pi}{4}$ and $x = \lambda$ both $-\pi \leq x, \lambda \leq \pi$.

Google uses x, y coordinates where $(x = 0, y = 0)$ is the point at the international date line $\lambda = -\pi$ and $Y = \pi R$ which is equal to half the circumference of the earth. Then, of course, the point $(x = 1, y = 1)$ coincides with the $\lambda = \pi$ and $Y = -\pi R$. The transformation from Mercator coordinates to google coordinates is straightforward:

$$x = \frac{X}{2\pi} + 0.5 = \frac{\lambda}{2\pi} + 0.5 \quad (3)$$

$$y = \frac{1}{2}\left(1 - \frac{Y}{\pi}\right) \quad (4)$$

$$y = 0.5\left(1 - \frac{1}{2\pi}\ln\left(\frac{1+\sin(\phi)}{1-\sin(\phi)}\right)\right) \quad (5)$$

The π in the y equation makes sure that it the inner expression varies from 0 to 1 when Y varies from π to -π. Multiplying by 0.5 makes sure that y varies from 0 to 1 when \phi varies from north to south starting and ending at the latitudes where $|Y| = \pi$, the choice that Google made for the top and bottom limits of the maps.

The backward transformation is also possible. With Y=$\pi(1 - 2y)$

$$\lambda = X \quad (6)$$

$$= 2\pi x - \pi \quad (7)$$

$$\phi = 2\,\mathrm{atan}(\exp(Y)) - \frac{\pi}{2} \quad (8)$$

$$= 2\,\mathrm{atan}(exp(\pi(1 - 2y))) - \frac{\pi}{2} \quad (9)$$

The value of ϕ where $y = 0$, i.e. where $Y = \pi$ is a little above 85 degrees. This is therefore also the top of the earth as far as Google Maps is concerned. Google Earth is a completely different matter. In Google Earth at every moment the current view is projected on the screen of the computer.

See also [Marcator (2011)] and [Savage (2006)]. I found it hard to find the exact details on the web. It seems that many have some application or code or just use the code provided by Google Map API, yet explantion of the background is hard to find. Here it is.

Figure 2 shows what y becomes by straight forward projection of points on the face of the earth onto the Mercator cylinder (with radius 1) and after transformation of y to make the map conformal. The differnce is material only at higher latitudes although both go to infinity ay $N = \pm 90^o$

Here is the code that generated the figure.

```
phi = 0:0.001:(85/90)*pi/2;
y1  = tan(phi);
y2  = 0.5*log((1+sin(phi))./(1-sin(phi)));

figure; hold on;
plot(180/pi*phi,y1,'r');
```

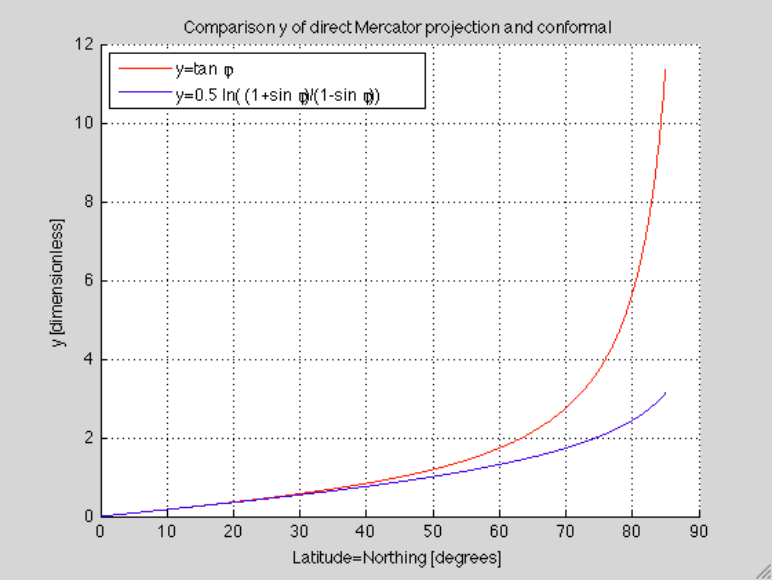

Figure 2: Comparison of the y when direcly projected on the Mercator cylinder (y=tan(phi) and after transformation to make the Mercator map conformal.

```
plot(180/pi*phi,y2,'b');

legend('y=tan \phi','y=0.5 ln( (1+sin \phi)/(1-sin \phi))',2);
grid on
xlabel('Latitude=Northing [degrees]');
ylabel('y [dimensionless]');
title('Comparison y of direct Mercator projection and conformal');
```

It is a nice exercies to plot both projections of small circles on the globe, which shouls show that the mercator tranformation will not distort them while direct projection does and more so toward the poles.

10.4 Google Maps tiling coordinates

From tis point onward, the coordinate system used by Google Maps is surprisingly simple. It allows to immediately use coordinates in units like meters within any retrieved figure.

In Googles x, y coordinates the world is flat. It is use to project any map on any zoom level. A each zoom level each map is divided inton 4 submaps of equal size in x, y coordinates. At zoom level zero the entire world (except the poles) fits in one figure with x,y coordinates between zero and one. The Lat=0, Lon=0 point then coincides with $x = 0.5, y = 0.5$. At zoom level 1 this point forms the lower right of the first tile and the upper left of the last, the fourth tile. And so on.

The width and height of every tile in x,y coordinates at zoom level n are the same and equal to

$$N = 2^n,\ w = 1/N$$

where n is zoom level and N the zoom factor or the number of tiles in both E-W as N-S directions. Google maps currently provides 21 zoom levels. This corresponds at the equator in EW direction with 19.0879 m using for the radius of the earth R2 = 6371007.2 m, whcih is the earth's authentic radius (Geodetic Union) and in north south direction with 9.5439 m. Because the projection is stretched near the globes, the resolution gets better with lattitude.

To work with tiles and zoom levels you need four numbers, i.e., 2, ix, and iy, to denote the hoirzontal and vertical number of the tile and two px and py to fix the position within the tile, and, finally, one number to store the zoomlevel itself. These five numbers have to be kept strictly together at all times in order not inadvertently find your self on a wrong tile!

Google divides the tile in 256x256 pixels and, therefore, the pixels dfined the final resulution and define the coordinates within the tile uniquely. The pixel coordiantes px and py are also relative to the upper left corners of the tile. This is the case within any zoom level. Therefore the highest possible resulution on any Google

Maps image near the equator will be 0.0746 m in EW direction and 0.0373 m in NS direction sufficient for most applications.

10.5 How to determine the tile and local coordinates of Google Map figures

The tile width w is 1 at zoom level 0. At any zoom level n it becomes

$$w_n = 1/N, \text{ with } N = 2^n \tag{10}$$

N is the number of tiles spanning the globe in both directions at zoom level n. Hence, w starts with 1 and becomes smaller and smaller the more tiles span the globe. The tile coordinate at zoom level n thus becomes

$$\begin{aligned} x_n &= x/w_n & (11)\\ y_n &= y/w_n & (12) \end{aligned}$$

The integer part corresponds to the tile number in EW and NS direction respectively

$$\begin{aligned} i_{xn} &= \text{fix}\,(x_n) & (13)\\ i_{yn} &= \text{fix}\,(y_n) & (14) \end{aligned}$$

The fractional parts correspond with the relative coordinates within the tile. The Google Maps pixel coordinate within the tile, therefore, becomes

$$\begin{aligned} p_{nx} &= 256\,(x_n - \text{fix}\,(x_n)) & (15)\\ p_{ny} &= 256\,(y_n - \text{fix}\,(y_n)) & (16) \end{aligned}$$

All these coordinates are unique in terms of (E N) and using the radius of the earth distances van be computed within every GM figure.

10.6 Back transformation from Google Maps to Lat Lon

The other way around, we need the tile in which our position falls and whch corresponds with the zoom level:

$$\begin{aligned} x &= w_n\,(i_{nx} + p_{np}/256)\\ y &= w_n\,(i_{ny} + p_{np}/256) \end{aligned}$$

And, finally

$$\begin{aligned} E &= 360x - 180 & (17)\\ N &= 90 - 180y & (18) \end{aligned}$$

by setting px and py to 0 and 256 respectively one can immediately find the E and N coordinates of the corners of the file. With these coordinates any image can be read in with Easting on the horzontal and N on the vertical axes. For instance, say you have compute the limits E1 and E2 and N1 and N2 of the tile, then

```
N=52.372821; E=4.893667; % Dam Monument Amsterdam
zoom=16; figsize=[512,512]; % in Google pixels
url.center = [N,E];         % Lat Lon
url.zoom   = zoom;          % between 0 and 21
url.size   = [512 512];     % each between 1 and 640

URL=mf_GM2PIC(url);  % URL to get the GM picture
```

```
[ix,iy,px,py]=mf_GMLL2pix(N,E,zoom);  % picture center in Google Coords

east  = [py-figsize(1)/2 py+figsize(1)/2];
north = [px+figsize(2)/2 px-figsize(2)/2];

NLim = mf_GMpix2N(ones(2,1)*ix,ones(2,1)*iy,north);
ELim = mf_GMpix2E(ones(2,1)*ix,ones(2,1)*iy,east);

A    = imread(URL);          info = imfinfo(URL); %
image(ELim,NLim,A); colormap(info.colormap);
```

10.7 Coordinates in meters within a tile

This is just a matter of computing the real world coordinates inside the tile. The most obvious way wil be to do that relative to the center of the tile. But it can be done relative to any point, for instanc the lower left corner of a convinient recognizible point in the picture. All we need is the EN coordinates of the point relative to which the coordinates will be computed. Of course, the real-world coordiantes will vary in scale within any figure, due to its projection, whereas this is not the case if we use EN coordinates directly. To remain fair in accuracy we can apply the scale of the center of the tile and use that to relate all other points of the figure. This ensures that the vertical lines in the figure remain vertical. When the scale is not very large, this method is accurate enough for most practical purposes. If not revert to N-E (Lat Lon) coordinates.

The size of the tile at its center can be computed using equations 19 and 20

$$\frac{\partial \lambda}{\partial x} = 2\pi \quad \rightarrow \quad \Delta\lambda = 2\pi w \tag{19}$$

$$\frac{\partial \phi}{dy} = \frac{1}{\cosh(y+\pi)} \quad \rightarrow \quad \Delta\phi = \frac{w}{\cosh(y+\pi)} \tag{20}$$

=

$$\frac{\partial \phi}{\partial y} = \frac{1}{\cosh(Y)}$$

$$\frac{\partial \phi}{\partial y} = -\frac{2\pi}{\cosh(\pi(1-2y))} \tag{21}$$

And from this

$$\begin{aligned} \Delta X_w &= \Delta\lambda R\cos(\phi) \\ \Delta Y_w &= \Delta\phi R \end{aligned}$$

Example

```
% Continued from the previous example

w   = 2^(-zoom)
phi = pi*N/180-90; % Easting is immaterial here
y   = 0.5*log((1+sin(phi))/(1-sin(phi)))-pi;

DXtile = 2*pi*w*R*cos(phi);  % width  of tile
DYtile = w/cosh(y+pi)        % height of tile
```

To get the actual tile size see function
[dx dy]=mf_GM_tile_size(zoom,Lat)

```
[dx dy] = mf_GM_tile_size(zoom,Lat)
```

10.8 GM figures centered around arbitraty locations not a tile origin

Once we have the figure axes in the correct size and limits, we can plot directly on them in the coordinates of those axes. You may try this example. It constructs the struct url which is passed to URL=mf_GM2PIC(url) to get the complete URL string for the request. It may be put in the browser to the the figure in the brouwser. Yuo can directly get the figure in the browser by leaving out the URL output argument. Instead of presenting the URL string it will put it in the browser for you.

Here we want the URL to allow getting the Google Maps figure directly in a Matlab figure so we can use it in our model. Thefore, we use A=imread(URL) to readit followed by info=iminfo(URL) to get addional information about the figure, such as its colormap. Then we show it and use the axis in meters corresponding to minus and plus half the figure width, so that the axes are centered at the center of the figure.

Finally a 600 by 600 m grid is plotted onto the figure, so that you can verify the scale.

The script has some other locations that you can try immediately. You can also change the zoom level between 0 and 21.

```
% TESTURL:  Requests a map directly from Google Maps into Matlab figure
%
% USAGE:
%   testURL
%
% DETAILS:
%   The functions can be found in mflab/mfiles/visualization
%
% TO 110506

clear variables; close all;

basename='testURL';

%% Coordainats of an island from Dubai World
%url.center=[ 25.253159, 55.178000];   % Dubai world island
url.center=[ 52.372821,  4.893667];    % Dam Monument Amsterdam
%url.center=[ 68.962970, 33.089563];   % Murmansk
%url.center=[-33.856857,151.215192];   % Sydney
% url.center=[-54.795444,-68.232218];  % Ushuaia, Argentina
% url.center=[ 37.808810,-122.409803]; % San Francisco, Fisherman's Wharf
% url.center=[ 40.748524,-73.985676];  % New York Empire State building
url.zoom=16;             % must be between 0 and 21
url.size=[512 512]';     % must be < 640 figure pixel size
url.maptype='satellite';

[dxTile dyTile]=mf_GM_tile_size(url.zoom,url.center(1)); % tile size in m
xLim=dxTile*url.size(1)/256*[-0.5 0.5];
yLim=dyTile*url.size(2)/256*[-0.5 0.5];

%% the picture
URL  = mf_GM2PIC(url);              % URL to get the GM picture
A    = imread(URL);                 % get it directly into matlab
info = imfinfo(URL);                % get associated image info

image(xLim,yLim([2 1]),A);
colormap(info.Colormap);            % its colormap from info
hold on;

set(gca,'ydir','normal');
xlabel('x [m relative to tile center]');
ylabel('y [m relative to tile center]');
title('test to get google figure');
```

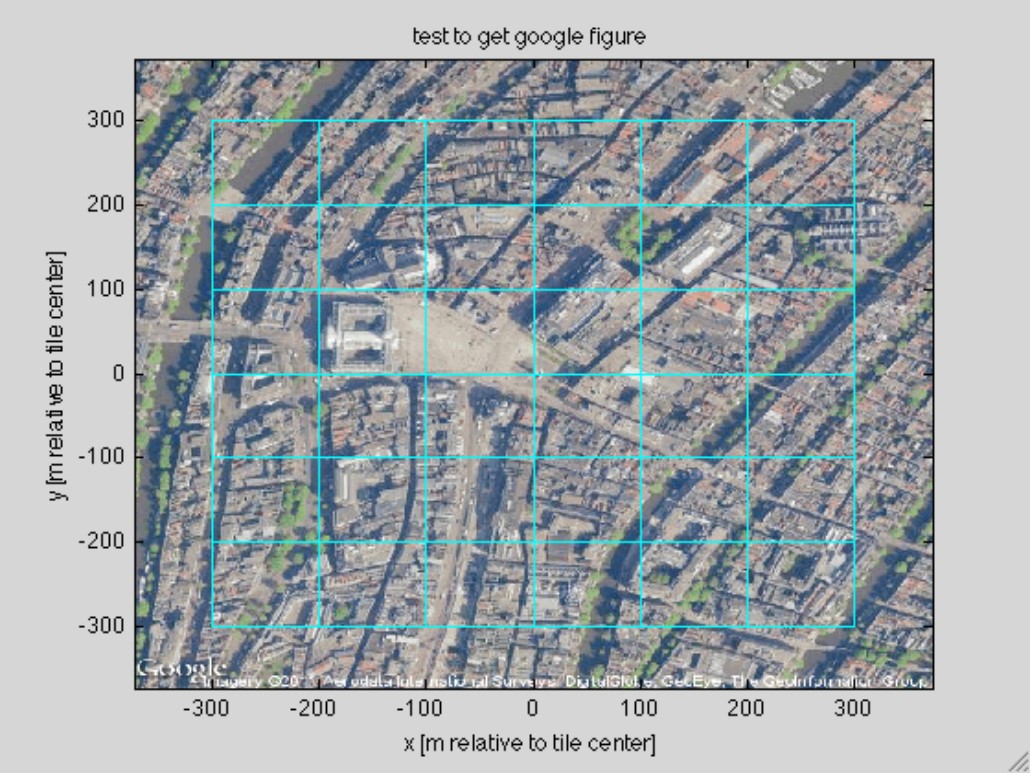

Figure 3: Result of testURL, showing the requested figure with Amsterdam national monument at its center and overlain with a 600m by 600m grid with 100 m spacing in both directions (zoom level 16).

```
%% Grid
xGr=-300:100:300;
yGr=-300:100:300;

[xGr,yGr,xm,ym,DX,DY,NX,NY]=modelsize(xGr,yGr); % Coordinate housekeeping

plotgrid(xGr,yGr);
```

Here is an example in which N and E (lat and lon) are used as axes. It uses the function mf_GM_tile_size with an extra argument to singnal that $[dE, dN]$ are required as output instead of $[dx, dy]$. it uses the actual N and E on the axes and plots a grid with 1 degree spacing around the center of the figure.

```
% TESTURL2:  Requests a map directly from Google Maps into Matlab figure
%
% USAGE:
%   testURL
%
% DETAILS:
%   The functions can be found in mflab/mfiles/visualization
%
% TO 110506

clear variables; close all;

basename='testURL';

%% Coordinates of an island from Dubai World
%url.center=[ 25.253159, 55.178000];    % Dubai world island
```

```
url.center=[ 52.372821,  4.893667];      % Dam Monument Amsterdam
%url.center=[ 68.962970, 33.089563];     % Murmansk
%url.center=[-33.856857,151.215192];     % Sydney
% url.center=[-54.795444,-68.232218];    % Ushuaia, Argentina
% url.center=[ 37.808810,-122.409803]; % San Francisco, Fisherman's Wharf
% url.center=[ 40.748524,-73.985676];  % New York Empire State building
url.zoom=5;              % must be between 0 and 21
url.size=[512 512];      % must be < 640
url.maptype='satellite';

%% Tile pixel limits of picture

[dETile dNTile]=mf_GM_tile_size(url.zoom,url.center(1),'NE');

ELim=dETile*url.size(1)/256*[-0.5 0.5]+url.center(2);
NLim=dNTile*url.size(2)/256*[-0.5 0.5]+url.center(1);

%% the picture
URL  = mf_GM2PIC(url);                % URL to get the GM picture
A    = imread(URL);                   % get it directly into matlab
info = imfinfo(URL);                  % get associated image info

image(ELim,NLim([2 1]),A);
colormap(info.Colormap);              % its colormap from info
hold on;

set(gca,'ydir','normal');
xlabel('E^o [m relative to tile center]');
ylabel('N^o [m relative to tile center]');
title('Figure in NE + 1 degree grid around central point');

%% Grid
EGr=-10:1:10;
NGr=-10:1:10;

[EGr,NGr,Em,Nm,DE,DN,NE,NN]=modelsize(EGr+url.center(2),NGr+url.center(1)); % Coordinate h

plotgrid(EGr,NGr);
```

The method exploited here is used in the example mflab/examples/swt_v4/FFSE/FSSE-HFB.

11 Modeling heat transport

Heat is treated mathematically the same a diffusion combined with sorption. Hence, we need to specify both the diffusion coefficient of the cells on a per layer or per cell basis as well as the sorption process. The diffusion coefficient is specified in the LAY worksheet on a per layer basis, which may be mixed with coefficients on a per cell basis, which is given as the parameter DMCOEF in mf_adapt. *mfLab* uses the mf_adapt values if DMCOEF in the LAY-sheet of a given layer has a negative value. See section 6 on page 6. Diffusion coefficients are handed over to MT3DMS and SEAWAT through the DSP (dispersion-diffusion) package and sorption coefficients through the RCT (reaction) package.

To model conduction-convection of heat with MT3DMS or SEAWAT, we must compare the mathematical formulations. If we are merely interested in pure diffusion and pure conduction, so no flow (no dispersion and no convection), there is no need for the reaction package. This can be shown as follows. The mass balance equation and heat balance equation are then, noting that in the left equation c is concentration and in the right equation c is heat capacity (!) :

$$\epsilon\left(1+\frac{\rho_b K_d}{\epsilon}\right)\frac{\partial c}{\partial t}=\epsilon D_s\nabla^2 c, \quad \rho c\frac{\partial T}{\partial t}=\lambda\nabla^2 T$$

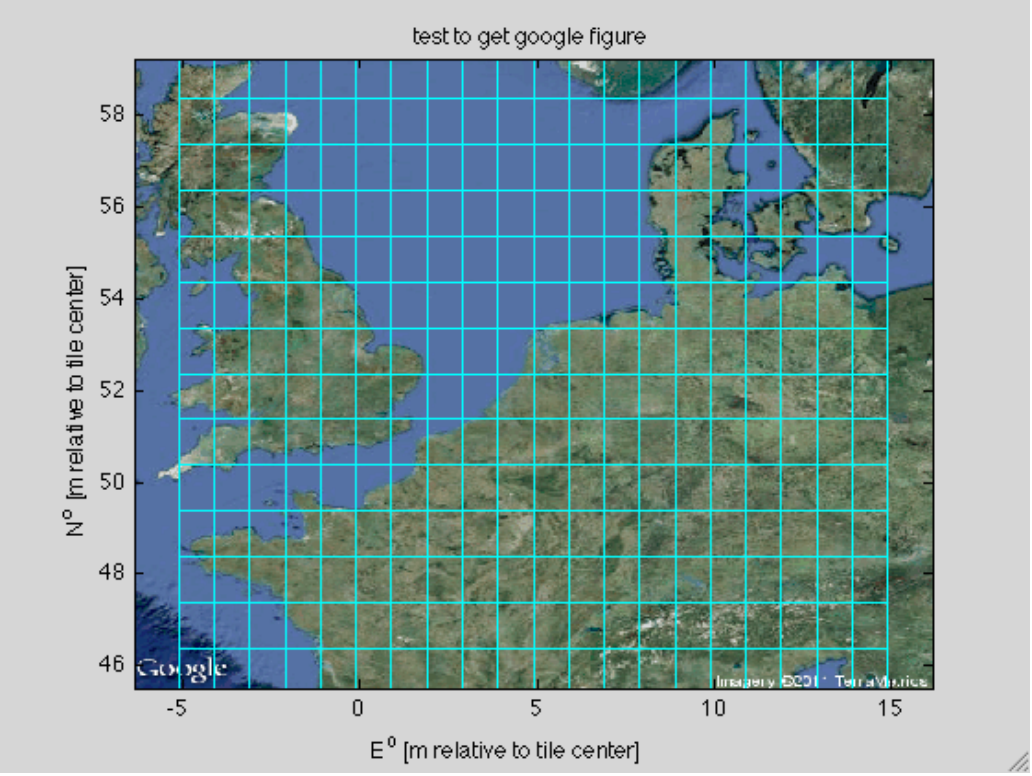

Figure 4: Result of of testURL2 with zoom level 5 (see mflab/mfiles/visualization). E and N (LON and LAT) are used on the axes, and a 1-degree grid overlay is plotted with center at Amsterdam (see code snippet for coordinates).

so that

$$\left(1+\frac{\rho_b K_d}{\epsilon}\right)\frac{\partial c}{\partial t}=D_s\nabla^2 c, \qquad \rho c\frac{\partial T}{\partial t}=\lambda\nabla^2 T$$

Hence, both systems are equivalent if we set

$$D_{\text{diff}}=\frac{D_s}{1+\frac{\rho_b K_d}{\epsilon}}=\frac{\lambda}{\rho c}$$

and simulate only diffusion, without sorption, or, equivalently, set $D_{\text{diff}}=D_s=\lambda/\rho c$, with $1+\rho_b K_d/\epsilon=1$, so that $K_d=0$. That is, use linear sorption but set the distribution coefficient equal to zero.

Alternatively, we may set

$$D_{\text{diff}}=D_s=\lambda, \quad K_d=\epsilon\frac{(\rho c-1)}{\rho_b}\simeq\epsilon\frac{\rho c}{\rho_b}$$

That is use the bulk heat conductance as diffusion coefficient and $\epsilon(\rho c)/\rho_b$ as distribution coefficient.

11.1 With flow

In the case we have flow so that advection (and, therefore, dispersion) and or convection are working, we have to include those processes in the heat and mass balance:

$$\epsilon\left(1+\frac{\rho_b K_d}{\epsilon}\right)\frac{\partial c}{\partial t}=\epsilon D\nabla^2 c-\epsilon v\nabla c, \quad \rho c\frac{\partial T}{\partial t}=\lambda\nabla^2 T-\epsilon\rho_w c_w v\nabla T$$

so that the left equation leads to

$$\frac{\partial c}{\partial t}=\frac{D}{R}\nabla^2 c-\frac{v}{R}\nabla c$$

Now we make the right-hand equation equivalent to the left hand one

$$\rho c \frac{\partial T}{\partial t} = \lambda \nabla^2 T - \epsilon \rho_w c_w v \nabla T$$

using $\rho c = \epsilon \rho_w c_w + \rho_b c_s$, and dividing left and right by \rho_w c_w, we get

$$\epsilon \left(1 + \frac{\rho_b c_s}{\epsilon \rho_w c_w}\right) \frac{\partial T}{\partial t} = \frac{\epsilon \lambda}{\epsilon \rho_w c_w} \nabla^2 T - \epsilon \frac{\rho_w c_w}{\rho_w c_w} v \nabla T$$

$$\epsilon \left(1 + \frac{\rho_b K_d}{\epsilon}\right) \frac{\partial T}{\partial t} = \epsilon D_H \nabla^2 T - \epsilon v \nabla T, \quad K_d = \frac{c_s}{\rho_w c_w}, \quad D_H = \frac{\lambda}{\epsilon \rho_w c_w}$$

so that

$$\frac{\partial T}{\partial t} = \frac{D_H}{R} \nabla^2 T - \frac{v}{R} \nabla T$$

So that equivalence is achieved when setting

$$D_H = \frac{\lambda}{\epsilon \rho_w c_w}, \text{ and } K_d = \frac{c_s}{\rho_w c_w} \text{ and } R = 1 + \frac{\rho_b K_d}{\epsilon} = 1 + \frac{\rho_b c_s}{\epsilon \rho_w c_w} = \frac{\rho c}{\epsilon \rho_w c_w}$$

As can be seen upon inspection of the right-hand expression, the retardation in the case of heat transport is the total heat capacity of a m3 of porous medium including water over the heat capacity of the water in the pores, i.e. the present water. This is always the case: the retardation is the total mass per m^3 of porous medium over the mass dissolved in the present water, the porosity times the concentation in the water. The equivalant distribution coefficient in the case of heat is now also known. It adheres exactly to the definition of the distribution coefficient. Namely, given water a certain tempeature, the the heat stored in a m^3 of this water is $\rho_w c_w$, while the heat stored in a kg of solids with the same temperature is c_s.

Also the equivalent diffusivity $D = \frac{\lambda}{\epsilon \rho_w c_w}$ can be physically understood. In diffusion/dispersion D is the total mass flux through the pores driven by the concentration gradient. In the case of heat \lambda is the total heat flux through both pores and solids. To make this heat flux comparible with the diffusion/dispersion case, then we do as if this heat flux is through the pores, which translates into a temperature flux equal to $\lambda/(\epsilon \rho_w c_w)$.

In simulating heat, the latter case is the most general, as it allows for groundwater flow to be combined with heat flow. To simulate heat with MT3DMS or SEAWAT we thus have to specify

$$D_{\text{diff}} = \frac{\lambda}{\epsilon \rho_w c_w}, \quad K_d = \frac{c_s}{\rho_w c_w}$$

11.1.1 Example

For ordinary value of the parameters we may set

$$\lambda = \epsilon \lambda_w + (1 - \epsilon) \lambda_s$$

assuming $\epsilon = 0.35$, $\lambda_w = 0.6\,\text{W/m/K}$ and $\lambda_s = 3\,\text{W/m/K}$, we have

$$\lambda = 0.35 \times 0.6 + (1 - 0.35)\, 3.0 = 2.16\,\text{W/m/K} = 0.19 \times 10^6\,\text{J/d/m/K} = 68 \times 10^6\,\text{J/y/m/K}$$

Bulk heat capacity is

$$\rho c = \epsilon \rho_w c_w + (1 - \epsilon)\, \rho_s c_s$$

with $\rho_w = 1000\,\text{kg/m}^3$, $\rho_s = 2650\,\text{kg/m}^3$, $c_w = 4200\,\text{J/kg/K}$, $c_s = 800\,\text{J/kg/K}$,

$$\rho c = 0.35 \times 1000 \times 4200 + (1 - 0.35) \times 2650 \times 800 = 2.85 \times 10^6\,\text{J/m}^3\text{/K}$$

$$D_{\text{diff}} = D_H = \frac{\lambda}{\epsilon \rho_w c_w} \simeq \frac{0.19 \times 10^6}{0.35 \times 4.2 \times 10^6} = 0.13\,\text{m}^2\text{/d} = 47\,\text{m}^2\text{/d}$$

$$K_d = \frac{c_s}{\rho_w c_w} = \frac{800}{4.2 \times 10^6} = 1.9 \times 10^{-4} = \frac{1}{5250}\,\frac{\text{J/kg}_{\text{solids}}}{\text{J/m}^3_{\text{water}}}$$

The dimension being the heat per kg solids versus the heat per m^3 water of the same temperature.

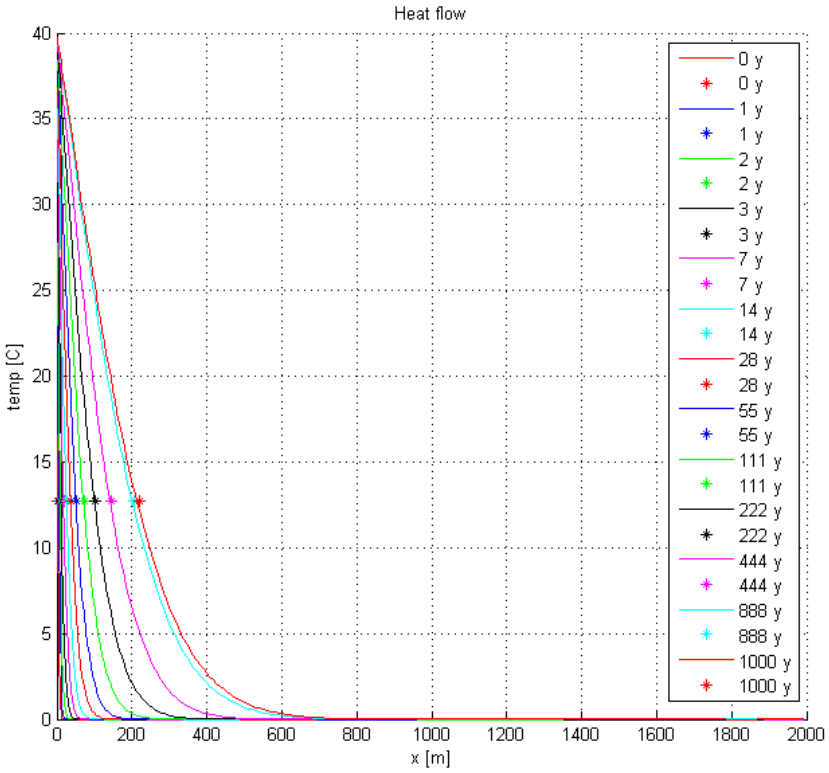

Figure 5: Heat conduction from a fixed temperature

For the retardation of the heat front we have:

$$R = \quad 1 + \frac{\rho_b K_d}{\epsilon} \quad = 1 + \frac{(1-0.35) \times 2650 \times 1.9 \times 10^{-4}}{0.35} = 1.94$$

$$R = \quad \frac{\rho c}{\epsilon \rho_w c_w} \quad = \frac{2.85 \times 10^6}{0.35 \times 4.2 \times 10^6} = 1.94$$

So that heat fronts travel with approximately half the velocity of water.

In the model we have to set $D_H = 0.13\,\mathrm{m}^2/\mathrm{d}$ in the LAY worksheet and $SP1 = K_d = 1.9 \times 10^{-4}\,\mathrm{m}^3/\mathrm{kg}$ in the LAY spreadsheet under column SP1. In the MT3D worksheet we need to specify linear adsorption through the parameter ISOTHM. For linear sorption $ISOTHM = 0$ and $SP1 = K_d$ while SP_2 is read but not used.

FIgure 5 shows the results of heat conduction form a constand temperature source at $x = 0$. The parameters used are as given above. The model was run in steady state, there is no groundwater flow, so pure conduction. The makers are the analytical solution results as

$$c = c_0 \mathrm{erfc}\left(\frac{x}{\sigma\sqrt{2}}\right)$$

which, for $\sigma = z$ always yields $c = c_0 \mathrm{erfc}(1/\sqrt{2}) = 0.32$, while $\sigma = x = \sqrt{2\frac{D_H}{R}t}$ was used to place these markers to verify the numerical results. The model was a single row model with 2000 cells in x-direction.

11.2 Die-out after an initial temperature profile

This situation can be simulated in at least two ways, one is to setup an initial concentration which will die out as time passes. To this end we used a start temperature equal to zero, except for distances smaller than W/2 (half aquifer width), where the start concentration was set equal to the initial temperture of 40C.

The analytical solution is

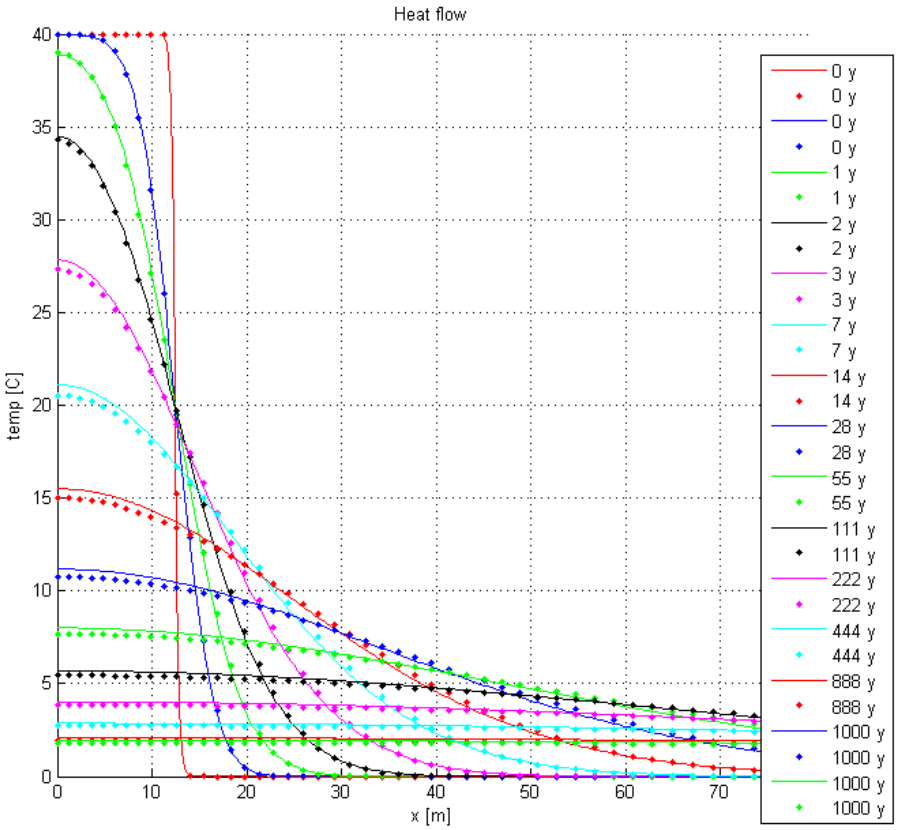

Figure 6: Die-out of an initial temperature of 40C (data used are given in the text)

$$T = \frac{T_0}{2}\left(\operatorname{erfc}\left(\frac{x - W/2}{\sigma\sqrt{2}}\right) - \operatorname{erfc}\left(\frac{x + W/2}{\sigma\sqrt{2}}\right)\right), \qquad \sigma = \sqrt{2\frac{D_H}{R}t}$$

The results are given in figure 6. The drawn lines are the results of the numerical computation with MT3DMS and the dots are the analytically computed values. There is a small deviation between the two for $x < W/2$ but which gradually disappears with time. I don't know the reason of this small deviation. It is not the grid accuracy because five times more grid points were used than points for the analytical compuation.

11.3 Sudden load of mass

Although this situation is clear, it is more tricky to solve numerically. Here we have to use two stress periods, one to inject the mass and a much longer one to let the concentration profile die out. Here the first stress period is 1 day and the second is 10 years, 36500 days.

The analytical solution for a initial mass M which dies off afterwards is given by

$$c = \frac{M}{\epsilon R}\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

To compare this mass with a concentration and an application width se set $M = \epsilon R c_0 W$. Transferrring the concentration to temperatures yields this:

$$T = \frac{\epsilon R T_0 W}{\epsilon R}\frac{e^{-\frac{z^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} = \frac{T_0 W}{\sigma\sqrt{2\pi}}e^{-\frac{z^2}{2\sigma^2}} \tag{22}$$

To compare with the previous case, we set $T^* = T_0(W/2)/dx = 40{\times}12.5/0.225 = 427\,\mathrm{C}$ as initial temperature in the first cell of the model, which has a width 0.225 m. This will store the same initial amount of heat in this

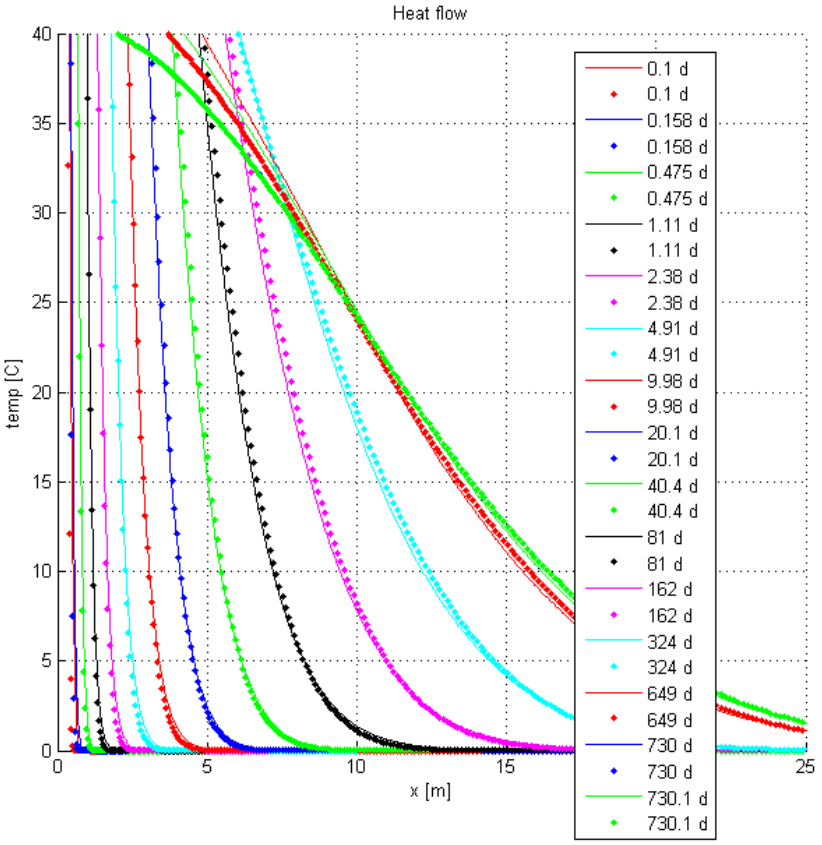

Figure 7: Result of heat pulse with the same total heat as in the case of an initial temperature over the width of the aquifer shown in figure 6

cell as previously stored in the half width of the aquifer at $T_0 = 40\,\mathrm{K}$. But the analytical solution remains as in 23, no model involved.

The results are given in the two figures below, one for short times after the release of heat and one for long times. The maximum temperature scale is set to 40C for easy comparison with the previous result.

Both figures show a good agreement with the analytical solution.

11.3.1 Injection pulse with mass loading

The last example is a pulse injection at time zero with the same amount of heat as was the initial situation in the previous cases. But this time the injection is provide by means of a mass loading, a direct injection of heat. This option can be used by setting ITYPE=15 in the point sources specified for the SSM package of MT3DMS.

$$T = \frac{M}{\epsilon R} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{z^2}{2\sigma^2}} \tag{23}$$

The model solves equation 23 and thus provides for division by ϵR. The M in terms of temperature equals

$$M = \epsilon R T_0 W$$

and the mass loading such that during the first stress period this mass is injected in the first cell of the model is

$$\frac{dM}{dt} = \frac{\epsilon R T_o W}{dt}$$

with dt the length of the first stress period.

This situation is case 5 of the example. The results are shown in the figure 9

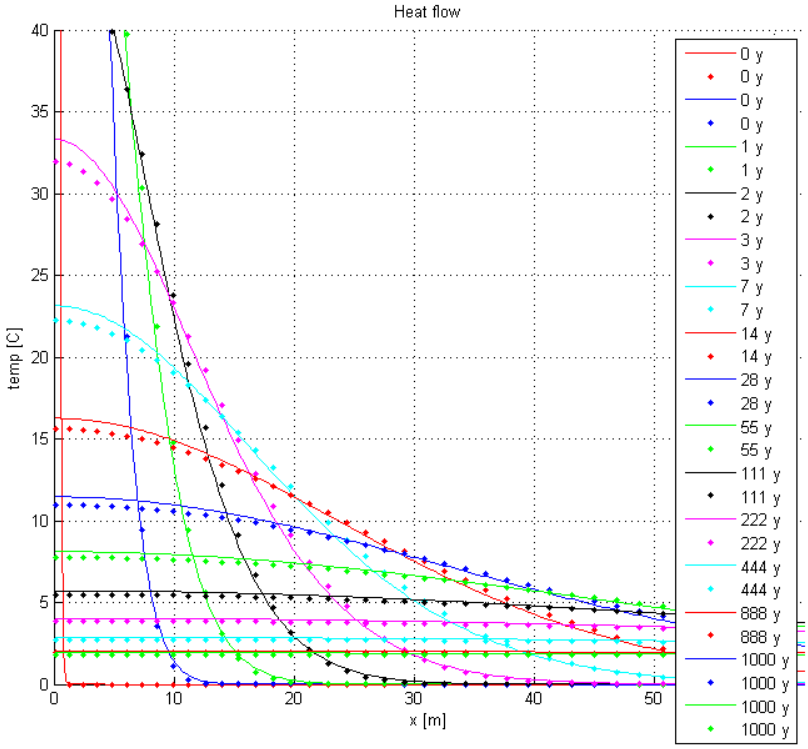

Figure 8: Result of heat pulse with same total heat as in the case of an initial temperature of 40C over the width of the 25 m thick aquifer in figure 6

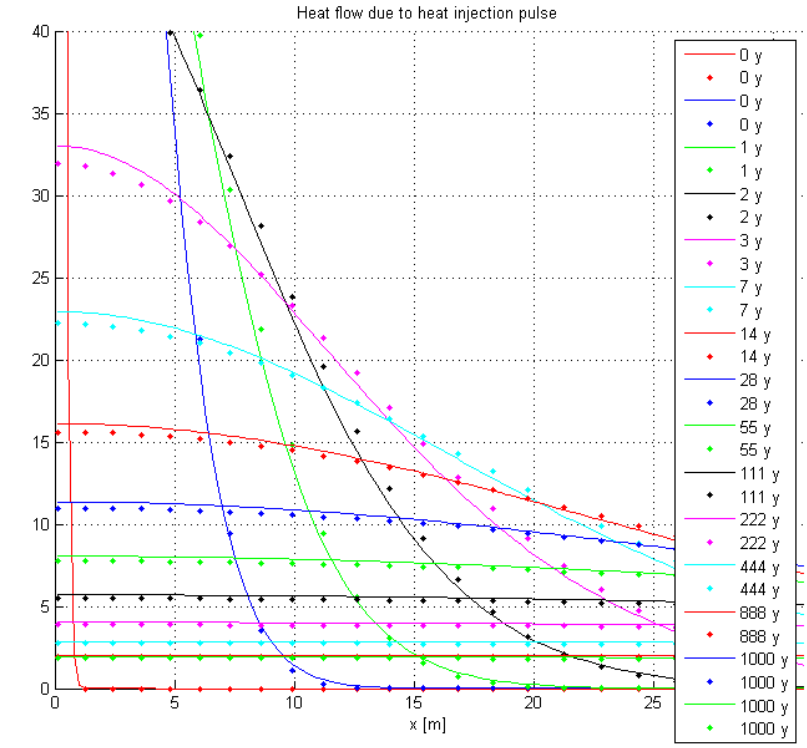

Figure 9: Temperatures after a sudden injection of heat at t=0 in the first cell of the model using mass loading (ITYPE=15 in the SSM package).

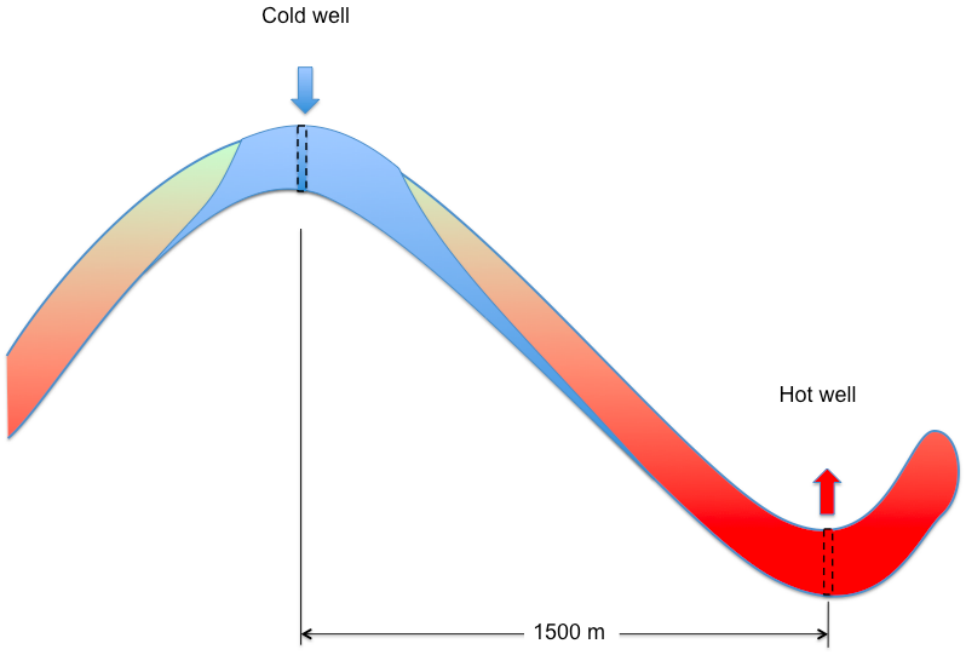

Figure 10: Impression of a geothermal aquifer (x-section) with extraction and injection well and spreading of cooled water subject to density and viscosity effects

11.4 Reheating of a geothermal system

Geothermal systems are claimed to be a sustainable future promise. Such systems extract hot groundwater from an appropriate depth, use the heat and inject the cooled water back into the same aquifer (layer) at a suitable distance, such that the cooled water does not reach the hot extraction well during the projected lifespan of the system. The heat is the temperature due to the normal geothermal gradient of about 30K/km. Hence at a depth of around 2 km, temperature in the order of 70C may be expected.

In favorable circumstances, these the temperature may be higher. The distance between the hot and cold well may be in the order of 2000 m, the thickness of the layer, often a sandstone about 20% and its thickness in the order of 100 m. A suitable layer may bend up- and downward under past tectonic movements in the earth's crust (figure 10). In such cases it is favorable to extract from the deeper, hotter, elevation and re-inject into the higher elevation to save drilling cost. The flow between the two wells is subject to heterogeneities and possible faults in the crust and layer. But this flow is also subject to viscosity effects, as the cooled water has a much higher viscosity than the original hot water, and it will be subject to density effects as the cooled water has a higher density than the hot water.

These effects make the flow complicated and careful study of the properties of the subsurface layers and flow processes are necessary for a good and safe design of a geothermal systems. We pass over all such detail here and ask ourselves how sustainable geothermal systems are. That is, how long does it take for the layers from which the heat was extracted until they are reheated again naturally and can be reused. Is this 1, 10 , 100, 1000 or 10000 years?

The cold front spreads out from the cold well to finally reach the hot well. After the cold front has passed a point in the aquifer, the adjacent over- and underlying layers will be cooled by the "cold" water in the geothermal aquifer (see figure 11). The duration of this cooling depends on the time since the passing of the cold front. Therefore, at the end of the lifetime of the geothermal system, adjacent layers near the injection well have been subject to cooling during the entire lifetime of the system, while near the front this cooling time is zero. The figure given an impression of this situation. It shows the temperatures in the geothermal aquifer between the injection cold well and the extraction hot well and also the cooling of the adjacent layers above and below. Hence, at the end of the lifetime or life-cycle of a geothermal system, for any point there is a certain time since the cold front passed and cooling of the adjacent layers has been proceeding, additionally the temperature in the geothermal aquifer is equal to the injection temperature. We will answer the question how long reheating takes for such a point.

Reheating is, in fact not a good concept. What happens is that the temperature anomaly due to the injection

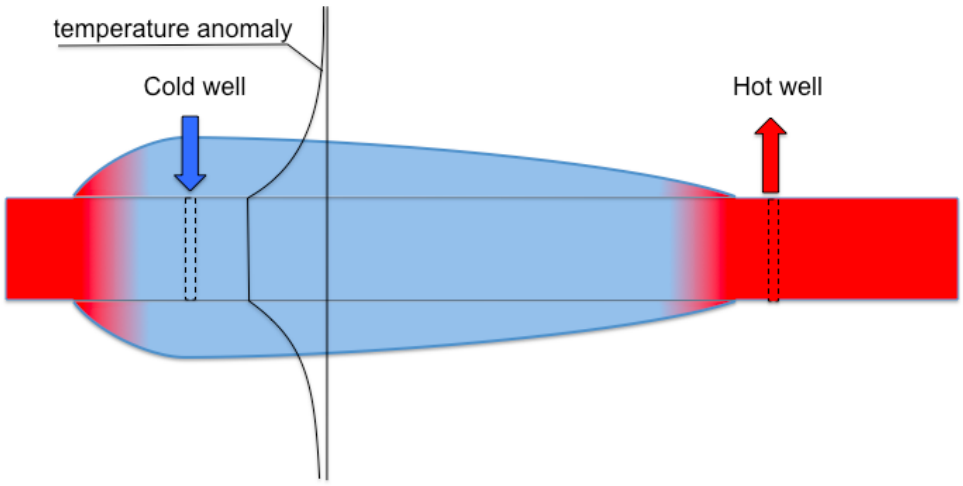

Figure 11: Geothermal aquifer (x-section) with cooled injection fluid moving towards the extraction well, while also cooling the overlying and underlying layers. Effects of density and viscosity ignored.

of relatively cold water is superimposed upon the natural initial temperature, at least during the time that the boundary temperature at the surface of the earth plays no role. That is the time during which the temperature anomaly does not reach ground surface. As this time is very long, it may be neglected at first, only to be checked later. If the effect of the temperature boundary at ground surface is negligible on the temperature distribution around our geothermal system, then reheating is not the right word. What takes place is a redistribution of the anomalous temperature under heat conduction, if, as we do here, influence of groundwater flow in these over- and underlying layers can be neglected. If they are not highly permeable, this is generally true.

Hence we consider the loss of energy to (gain of heat from) the layers above the geothermal aquifer, considering its top as a constant temperature layer during the time between the passage of the cold front and the end of the life of the system. The same is valid for the layers below the geothermal aquifer. Then, given the heat distribution thus obtained at the end of the system's life, we compute the dissipation over time, also taking into consideration the temperature anomaly in the aquifer itself, which dissipates from the point onwards.

Because of the principle of superposition we deal with only the temperature change $T_0 = 50C$ of the injection water compared to the original groundwater in the geothermal layer. The loss of heat into overlying or underlying layers from a constant temperature source follows from

$$T = T_0 \text{erfc}\left(\frac{z}{\sigma\sqrt{2}}\right), \quad \sigma = \sqrt{2\frac{D_H}{R}t}$$

in which

$$D_H = \frac{\lambda}{\epsilon\rho_w c_w}, \quad R = 1 + \frac{\rho_b c_s}{\epsilon\rho_w c_w} = \frac{\rho c}{\epsilon\rho_w c_w}$$

$$\lambda = \epsilon\lambda_w + (1+\epsilon)\,\lambda_s$$

$$\rho c = \epsilon\rho_w c_w + \rho_b c_s = \epsilon\rho_w c_w + (1+\epsilon)\,\rho_s c_s$$

The heat loss can be compute from this analytical solution as follows:

$$q_H = -\lambda\frac{\partial T}{\partial x} = \frac{\lambda}{\sigma}\sqrt{\frac{2}{\pi}}e^{-\frac{z^2}{2\sigma^2}}$$

At z=0, where the temperature is fixed, the total heat lost (or gained) is

$$H = \lambda T_0\sqrt{\frac{2}{\pi}}\int_0^t \frac{1}{\sqrt{2\frac{D_H}{R}}}t^{-1/2}dt$$

$$H = \lambda T_0\sqrt{\frac{2}{\pi}}\frac{2\sqrt{t}}{\sqrt{2\frac{D_H}{R}}} = T_0\sqrt{\frac{2}{\pi}}\frac{2\lambda t}{\sigma} = \epsilon\rho_w c_w R T_0\sqrt{\frac{2}{\pi}}\frac{2\frac{D_H}{R}t}{\sigma} = \rho c T_0\sigma\sqrt{\frac{2}{\pi}}$$

$$\frac{H}{\rho c} = T_0\,\sigma\sqrt{\frac{2}{\pi}}$$

where time is encapsulated in σ. Further, $H/\rho c$ is an equivalent thickness containing the same energy at $T = T_0$ as does the real system specified by σ. H is the total heat lost into the adjacent layers and, therefore, also the amount of (anomalous) heat present in these layers between the boundary and infinity.

The total amount of anomalous heat at this point of the geothermal aquifer between $\pm\infty$ equals

$$H_T = 2H + \rho c T_0 W = \rho c T_0 \left(2\sigma\sqrt{\frac{2}{\pi}} + W\right) \tag{24}$$

If t in σ equals the time between the passing of the cold front and the end of the life of the geothermal system, then H_T is the total amount of anomalous heat stored as this point in the aquifer between $-\infty \leqq z \leqq \infty$, which will dissipated after the system has been abandoned. Thus, σ in equation 24is a fixed value after the system stopped, we write further σ_0 for it.

Dissipation of heat from a sudden source is given by the following analytical solution

$$T = \frac{M}{\epsilon R}\frac{e^{-\frac{z^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}, \quad M = \epsilon R T^* dx$$

Where T^*dx the given initial temperature and dx the width over with this temperature is specified. A true pulse is where $dx \to 0$ and $T^*dx = constant$. For very long times, the initial distribution of the heat around $z = 0$ is of little importance, the distribution will gradually approach a the bell shape of the Gaussian normal probability density function. Therefore, for long times, we may ignore this initial distribution and consider the entire amount of anomalous heat in equation 24 as a single pulse at $t = 0$.

Hence

$$T = \frac{T_0\left(\sigma_0\sqrt{\frac{8}{\pi}} + W\right)}{\sigma\sqrt{2\pi}} e^{-\frac{z^2}{2\sigma^2}},$$

And the temperature in the center of the aquifer thus becomes

$$\frac{T}{T_0} = \frac{\sigma_0\sqrt{\frac{8}{\pi}} + W}{\sigma\sqrt{2\pi}}$$

The reheating time of the geothermal system may be equated to the time it takes until $T/T_0 = 0.05$, so that 95% of the heat anomaly has disappeared by dissipation of the heat anomaly into the overlying and underlying layers:

$$\sigma = \left(\frac{T_0}{T}\right)\left(\frac{\sigma_0\sqrt{\frac{8}{\pi}} + W}{\sqrt{2\pi}}\right)$$

from which

$$t = \frac{\sigma^2}{2}\frac{R}{D_H}$$

We may model this process in mfLab using a column of cells from ground surface to somewhere deep below the geothermal aquifer. We may then compute the initial situation analytically exact, or as pulse containing all anomalous heat lost at this point since the cold front passed by. Either method is accurate after long times, say 5 to 10 times the life span of the system. For visualisation purposes it may be nice to start with the analytical solution at the moment that the system is stopped and superpose this on the natural geothermal gradient. This will be done in this example.

The natural temperature gradient starts at say $T_0 = 10\,^{\circ}\mathrm{C}$ and increases by $G = -30\,\mathrm{K/km}$. Taking z upward positive and the center of the aquifer at Z_0,

$$T = T_0 - Gz$$

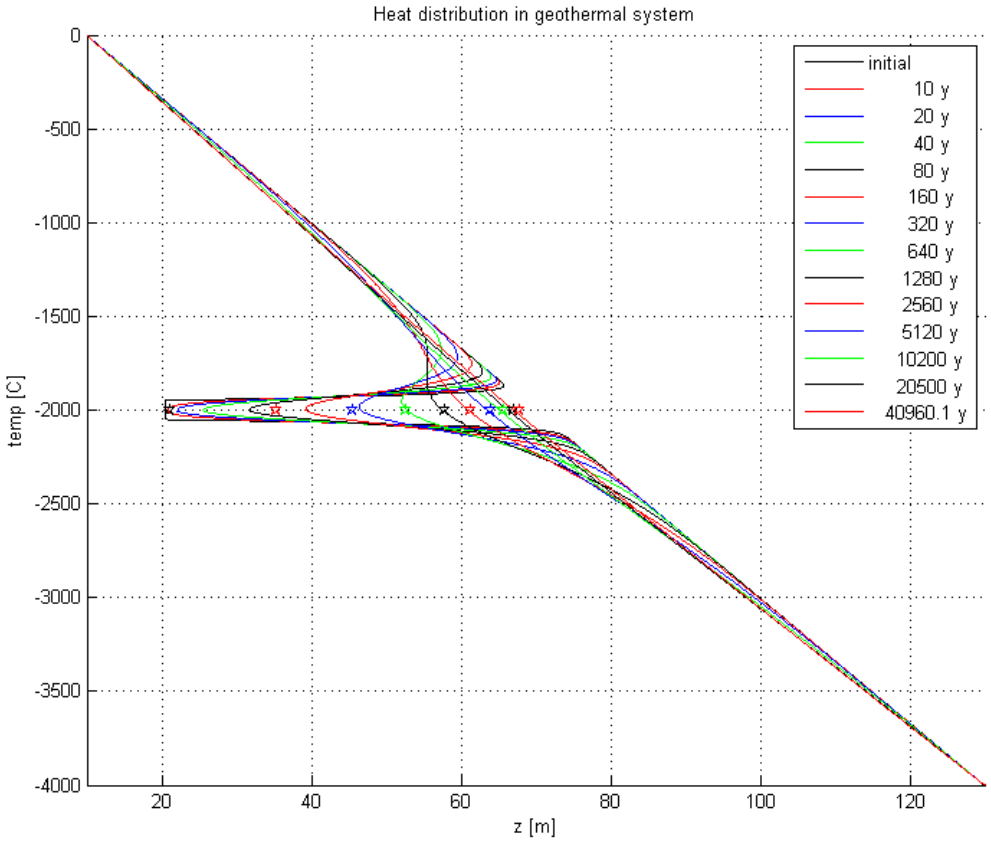

Figure 12: Temperature distribution and development after usage of a layer for geothermal heat extraction. The stars are the analytical solution temperatures at the center of the geothermal aquifer.

Between the top and bottom of the aquifer $Z_0 - W/2 \leqq Z \leqq Z_0 + W/2$ we have $T = T_{Z_0} - \Delta T$ and above and below we superimpose

$$\begin{aligned} \Delta T &= \Delta T_{\text{Top}} \operatorname{erfc}\left(\frac{z - (Z_0 + W/2)}{\sigma_0\sqrt{2}}\right), \quad z \geqq Z_0 + W/2 \\ \Delta T &= \Delta T_{\text{bot}} \operatorname{erfc}\left(\frac{-z + (Z_0 - W/2)}{\sigma_0\sqrt{2}}\right) \quad z \leqq Z_0 - W/2 \end{aligned}$$

With this initial temperature distribution we may compute the development over time using MT3DMS (or SEAWAT) with a single column of cells of $1\,\text{m}^2$ cross section. For convenience of plotting the y direction was chosen instead of z for this column. The column has 4000 cells in y direction. It is not feasible to make a model with 4000 layers instead. The input will then be much more extended and I'm not sure whether such a model will actually work. But a model consisting of a single column of 4000 cells in y direction was no problem at all for MODFLOW or MT3DMS.

The temperature at the top and bottom of the model have been fixed during this simulation. This is OK for the top but perhaps less so for the bottom. Nevertheless, the bottom is so far away from the geothermal aquifer that it will have no influence on the conclusions.

Note that this model has no groundwater flow, only heat conduction is taken into account.

The results are shown in figure 12, which demonstrates that reheating or rather the redistribution of the temperature anomaly caused by the use of the heat of a geothermal aquifer will take several tens of thousands of years in this case. The results of the analytical solution, i.e. the temperature at the center of the geothermal aquifer are also shown. Clearly, during the first years this solution does not match the numerical one because the initial temperatures differ a lot. But after about 300 years the two match accurately (figure 13). At the end the difference increases a bit due to the influence of the boundary conditions at the top and the bttom of

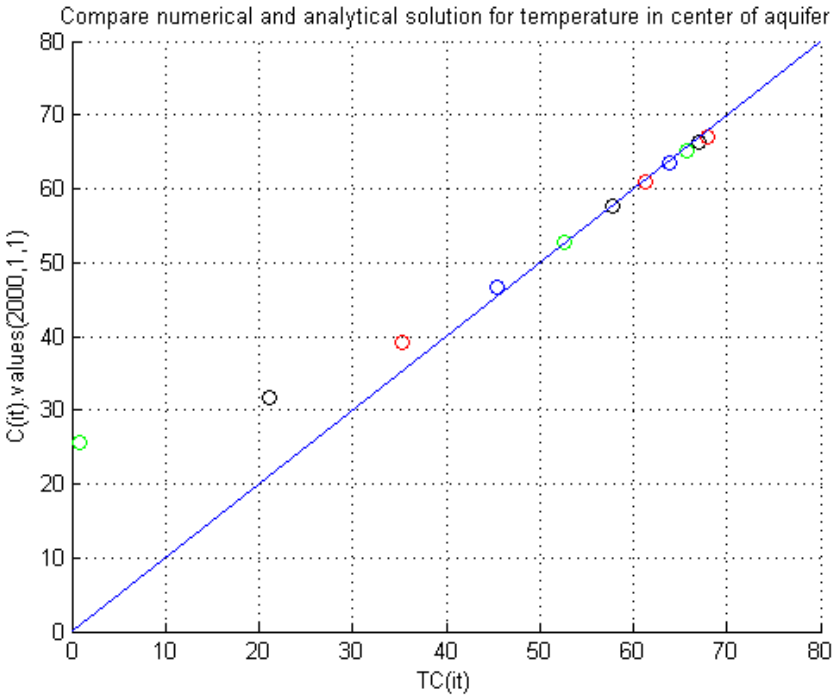

Figure 13: Comparison of the analytical and numerical soltuion for the temperature at the center of the geothermal aquifer. Small deviations at the end are due to influence of constant temperature boundaries. The points are the same as in the previous figure and span a period from 40-41000 years. The blue circle in the middel is 320 years.

the system, i.e. at 0 and 4000 m depth.

The time of reheating will be shorter if the layer is less thick, for instance several thousand years for a layer of 25 m thick instead of 100 m. With respect to the other parameters, i.e. heat capacity of solids and water nor heat conductance of solids and water, there will not be very much variation, at least no so much that this impression of the reheating time will be invalidated, except, perhaps, convective flows. But even these will take thousands of years.

12 Modeling heat loss from pipelines

Pipelines for fluid transport may be subject to temperature variations during the year and, especially also during between seasons. With regard to drinking water lines, temperature changes, become an increasing problem under the higher temperatures expected due to climate change. Next to that, parties become interested in using the thermal energy, either the "heat" or the "cold" present in water pipelines for their heating and cooling demand. In such situations, the heat exchange between the pipeline and the adjacent subsurface becomes of interest. In this example we model this heat exchange in *mfLab*.

The model consists of a pipeline of given radius completely filled with water and placed at a given distance below ground surface. Due to turbulence in the pipeline, the water temperature in the pipe is considered uniform. The influence of the pipe wall is ignored, we assume that this wall is highly conductive, for example, steel, which helps keeping the temperature inside the pipeline uniform, while, on the other hand, the pipewall does not hinder the heat exchange with the adjacent subsurface. The temperature will vary along the pipeline, but this is irrelevant when considering a cross section.

It is straightforward to model this heat transport with MT3DMS or SEAWAT, as was done in the previous example. However, if we want to compute the temperature along the pipeline using the information of a cross section, we need the heat loss as a function of time and the temperature in the pipe relative to that of the subsurface. The relation between de important factors may be established and quantified for a cross section and subsequently used to compute the temperature along the drain by a separate analytical or numerical approach.

To compute this heat flow, it is more convenient to simulate it as if the heat was groundwater. This can be

done with MODFLOW if we make sure the governing equations for groundwater and heat flow are mathematially equivalent.

The governing partial differential equation solved by the groundwater model is

$$S\frac{\partial\phi}{\partial t} = k\nabla^2\phi$$

while the equation governing heat flow is

$$\rho c\frac{\partial T}{\partial t} = \lambda\nabla^2 T$$

So that the comparison is almost trivial. We replace k by \lambda and S by \rho c and heat by temperature and use MODFLOW to compute the temperature and heat fluxes.

We may compute heat exchange and temperature effects for an arbitrary temperature profile in the pipe by convolution, once we derived the impulse-response or rather the step-response of the groundwater/ground temperature as a result of a sudden change of temperature in the pipeline. The step response can be readily computed using MODFLOW with the right replacement of groundwater quantities by those involved with heat tranport as explained. The impulse response is obtained as the derivative of the step response. The step response is obtained by a transient run with a single stress period in which the temperature at $t = 0$ is zero everywhere and unity in the pipeline. The result will then be the evolution of temperature with time in every point in the model as well as the evolution of all heat fluxes throughout the model. Most interesting for this analysis will be the total heat loss from the pipe as a function of time, which may also be computed as the total heat inside the model outside the pipe, as long as the heat change has not reached a boundary. This computation cannot be done analytically because this requires a mirror pipeline above ground surface, due to which it is not possible to maintain the constant temperature boundary at the pipe circumference.

13 Boundary conditions for flow

There are sveral packages by means of which boundary conditions may be specicfied to the flow problem: WEL, DRN, RIV, GHB, CHD. These package all require an input per stress period of the form

ITMP NP

Layer Row Col .....

where ITMP is the number of cells for which values are to be specified in the current stress period. Followed by ITMP lines of the given form, where .... stands for the specific cell input, which differs between the different types of boundaries. NP is the number of parameters used, which is generally zero.

If a given subsequent stress period has zero boundaries, than ITMP is zero. If in the next stress period the boundaries specified for the previous stress period are to be reused, then ITMP is -1.

In mfLab the boundaries may specified in the accompanying workbook (see spreadsheets WEL, DRN, GHB, RIV, CHD, which also shows the types of input required for each boundary type. The boundaries may also be specified directly in mf_adapt using a parameter with the corresponding names WEL, DRN, RIV, GHB, CHD. These parameters must hold an array in the form of a list where each line has the from

IPER Layer Row Col ...

mfLab will sort out the data and generate the correct MODFLW boundary file. In mfLab to specify a stress period with zero boundaries, just make sure there is no line with IPER equalling this stress period. mfLab will recognize this accordingly as "you don't want any boundaries specified for the corresponding boundary type in the missing stress periods".

In mfLab to specify that the stress period IPER should reuse the boundary specification of the previous stress period, just add onen line with -IPER (correct IPER but negative). It does not matter what layer row and column you specifiy next to this negative IPER on the same line. You may thus use all -1 or NaN or 0, mfLab just uses -IPER and ignores the other data on that line.

Example for WEL which requires Layer Row Col Injection flow

$$\begin{array}{rrrrr} 1 & 5 & 2 & 7 & -2400 \\ 1 & 3 & 3 & 8 & -1200 \\ -2 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 \\ 5 & 5 & 2 & 7 & -1200 \\ 5 & 3 & 3 & 8 & -1200 \end{array}$$

Lines 3 and 4 specifies that stress period 2 and 3 reuse the flows specified in lines 1 and 2 for the first stress period. IPER=4 is missing, so there are no injections/extractions in stress period 4, while new wells are specified in stress period 5.

The MODFLOW well input file produced by mfLab then yields, after some initial headings, the following stress period specifications:

```
2  0
5  2  7  -2400
3  3  8  -1200
-1  0
-1  0
0  0
5  2  7  -1200
3  3  8  -1200
```

Notice that in mfLab all stress period lines may be freely mixed as mfLab will sort them out using the IPER information in the first column. MODFLOW just requries the specifications to be provided in sequential order without explicit IPER information. Therefore, the MODFLOW input files requires strict order of the specifications and stress periods.

mfLab futher allows mixing the specification of the boundaries in both the accopanying workbook and directly in mf_adapt. If bouth cells with given discharge are defined in the worksheet WEl and as a parameter WEL in mf_adapt, mfLab will merge the info without checking for doubles. It uses the IPER information on each line of the WEL parameter in mf_adapt and in the accompanying worksheet to unite the info and attribute the info to the correct stress periods.

The input for the other types of boundaries DRN, RIV, GHB, and CHD works equally.

14 Boundary conditions for transport

The SSM package of MT3D requires boundary conditions for source-sink terms to be specified unless inflowing water is meant to have concentration zero. Hence boundaries with constant concentration and wells with given concentrations need to be specified. One has the option to use ICBUND to do so, but then these cells will behave as constant concentration cells throughout the simulation. This is a rather rare boundary condition for transport and, therefore not that often used.

For most sources and sinks concentration boundary conditions have thus to be specified. There is an option to do this in the workbook, worksheet PNTSRC (point sources). What is required is a list of

`PER LAYER ROW COL CSS ITYPE CSSMS_1 CSSMS_2 CSSMS_3`

for every cell that is a source or a sink.

where

```
PER = stress period
LAYER = layer number
ROW = row number
COL = column number
CSS = concentration of species in case only one species is used
ITYPE = type of boundary (fixed concentration, well etc)
ITYPE = 1 constant head cell
ITYPE = 2 is a well
ITYPE = 3 is a modflow drain
ITYPE = 4 is a modflow river
ITYPE = 5 is general head-dependent boundary cell
ITYPE = 15 is a mas loading cell
ITYPE = -1 is a constant concentration cell
CSSMS_1 .... are concentration of species 1, 2, 3 etc, as far as used. If more than one sp
```

Note that *mfLab* requires the stress period number as first item of the list, MT3DMS does not. However, requiring this number facilitates enormously the processing and frees the user of a burden, while it is much more secure. With the stress period number each line is unique and users may mix their input, for instance specifying all stress periods at once for each node instead of all nodes for each stress period at once before proceeding to

the following stress period. *mfLab* takes cae of sorting if necessary. Further, users are free to leave out data for stress periods. No sequential counting is involved.

When a large number of cells need to be specified, doing so in the spreadsheet is hardly an option. It will be much easier and more flexibly done in mf_adapt inside Matlab. mfLab has several functions to facilitate this, mainly ones that translate any part of a 3D cell array into a list as required by the boundary specification.

The function `indices=cellindices(I, dims, orderstr)` converts a list of global index numbers of an array with dimension dims=size(array) into a list of cell indices along the dimensions. For instance, we want to specify the PNTSRC required in the BTN package for the top of the model which has constant head. First get the global indices using Matlab's find function

```
Itop=find(Z>zm(1));
```

Where Z is supposed to be the 3D-array with top and bottom of all cells and zm(1) the elevation of the center of the topmost cell.

Then using the orderstring LRC to indicate we want layers, columns and rows in that order on each row of the cell index list

```
LCR=cellindices(Itop,size(IBOUND),'LRC');
```

Next set the stress period and boundary type numbers and the concentration (temperature) at the boundary

```
iSP=1; iType=1; TempTop=0;
```

Then generate a column of ones of length of I

```
u=ones(size(LRC(:,1)));
```

Then assemble the pointsource list

```
PNTSRC=[u*iSP LCR u*iType u*TempTop];
```

And that's it

The list can be extended with all kinds of other boundaries like

```
PNTSRC=[
    [u_1*iSP LCR_1 u_1*Temp_1 u_1*iType_1];
    [u_2*iSP LCR_2 u_2*Temp_2 u_2*iType_2]
    [...];
];
```

and so on.

Of course, these boundaries can also be read from a database. This way a PhD student reads in 635000 lines at ones and transfers these into a boundary list for input.

14.1 Constant concentration cells cannot be switched off, helas!!

The MT3DMS manual states for the SSM package that constant concentration cells ITYPE=1 cannot be switched off in subsequent stress periods once specified. It is possible to change the concentrations, however. From the point of usage this is a pity, because it is not possible to create an initial situation in one stress period and let it die out in the next periods. Such situation can, of course be computed using an intermediate step, i.e. first run a model to generate the wanted situation. Use those concentration as the start heads of the next run

15 Understanding Seawat input for viscosity and density

The input for the VDF and VSC modules in Seawat are flexible but terribly difficult to comprehend as result of the possible switches. After having spent in total several days wrestling with it, I attempted to make the description more easy to understand. Nevertheless, I hope that this input will be severely overhauled in the future so that people don't have to waste part of their remaining life time trying to figure out the tweaks of this way of specifying this input. I'm convinced it can be done more rigorously and straightforward as it still has some inconsistencies, especially with the options to read in density or viscosity data for specific stress periods and on the same time using the multi-species capabilities. These two are not compatible given the input structure.

One way is to include the logic-scheme of the input instructions (figure 15).

Understanding the logic of the VSC package (see Langevin et al. 2008, p20-21) has cost me many, many hours. I still think it's nasty. The most confusing is the logic that comes forth from the MT3DMUFLG. It's a three-way switch. If 0 then VSC is read in instead of computed. If >1 it is the number of a MT3D species used to relate viscosity with a concentration in a simple linear way. if -1 it is useful, as it allows using a sophisticated viscosity equation plus one or ore species to include their concentration in the viscosity equation. So if you

only want to use the more sophisticated viscosity-temperature equation use -1 with NSMEOS=0. In fact, you probably always want to use only the -1 switch for this reason.

15.1 Boundary conditions for constant head with variable density

Variable density boundary conditions can be somewhat complicated especially when the density changes during a simulation. The Seawat V4 manual on page 12-14 provides a clear explanation of the complexities and how to deal with them using the options provided by Seawat V4. The authors favor using CHD boundary package over ICBUND for given concentrations because CHD boundaries can vary during the simulation, for instance because of density changes. Instructions are given in on page 22. It's usage can be found in the mf_adapt of the examples/swt_V4/Coast.

To make the CHD package aware of the CHDDENSOPT it must be specified as a variable in mf_adapt like

```
CHDDENSOPT=2; % use environmental head at ocean boundary,
              % Langevin et al. 2008, p22
```

The value doesn't matter per se for the CHD package, but it can elegantly be used in the specification of the CHD input column where the CHDDENSOPT values has to be specified see below (6th column). If CHDDENSOPT is 1, an extra field CHDDENS is required. This can be done in the same way. Specify the variable and add its value as the right most (7th) column of CHD input.

```
..
LRCright=cellIndices(find(XM>xGr(end-1)),size(M),'LRC');
CHD=[];...
for iPer=1:NPER
CHD=[CHD;
[iPer*u LRCright u*[h_ocean h_ocean CHDDENSOPT]]
];
end
```

16 Steady-state versus transient flow with transport

One feature that often causes confusion is steady state of the flow model versus steady state of the transport model MT3MDS or SEAWAT. Even though the flow model maybe steady state, the transport model remains transient. Therefore, the time specified in the stress period for steady state periods matters for as far as the MT3DMS or SEAWAT are concerned. However, in case of a steady state stress period, the steps specified within that period don't matter. The flow model will compute the steady-state solution in a single step, wheras the transport model steps through time at the pace of its own transport steps, which are determined by the maximum permissible step size.

16.1 Viscosity in the NAM file with density package off

To use the viscosity package Seawat must run. But one may want to use viscosity without the density package on. mfLab is triggered to generate the input for Seawat, when it sees that the VDF package in the NAM worksheet is "on". Specifically to run Seawat without the density package on one may specify the on-switch for the VDF package on the NAM sheet as -1 instead of 1.

16.2 Density package

Figure 14 shows a mindmap of the input instructions of the Seawat V4 manual.

16.2.1 MT3DRHOFLAG ($\rho Flag$)

$\rho Flag$is the major 3-way switch in the density package. It can be -1, or >-1 (i.e. $\geqq 0$).

if $\rho Flag \geqq 0$ If $\rho Flag \geqq 0$ then then either the density is read in per stress period or it is computed with only one MT3DMS species is involved:

$$\rho = \rho_R + \frac{\partial \rho}{\partial c} c$$

There is no reference concentration included, which, therefore implies it is taken to be zero in Seawat if computed using item 4), where only ρ_R and $\frac{\partial \rho}{\partial c}$ are specified and no reference concentration c_R as is required in item 4c (see 26).

The manual says that if $\rho flag > 0$ it is the MT3DMS species number, however if it is zero, no MT3DMS species number is used or at least required by Seawat, as the density will be read in directly of through its concentration (25). It is not clear if and if yes which species number Seawat uses in case $\rho Flag = 0$.

$\rho Flag = 0$ **(reading density or concentration for each stress period)** if $\rho Flag = 0$, then non concentration species in involved and density will be read in or specified for each stress period according to the flag INDENSE.. This means that densities may bread for some stress periods while they may be computed for other stress periods. This flag INDENSE works as follows:

If INDENSE<0, the data from the previous period are reused or DENSEREF if the first stress period.

If INDENSE=0, set all to DENSEREF

if INDENSE >0, read item 7 (DENSE or CONCENTRATIONS) for that stress period.

if INDENSE=2, concentrations are read and converted to densities internally.

Directly reading of cell-density values will be rare. Its most likely application is a restart from a previous run.

- Items that are needed per stress period are specified in *mfLab* in the PER worksheet column "INDENSE" of the workbook for the problem on hand. If INDENSE is 1 for a stress period, then *mfLab* expects to find the specification of the densities to be read in the workspace parameter DENSE which must be a cell array with the cell corresponding to the stress period for which INDENSE==1 holding the 3D array with density values for all cells of the model.

$\rho Flag < 0$, $(\rho Flag = -1)$ **density computed using any series of species** If $\rho Flag < 0$, Seawat will compute density using NSRhoEOS (zero or more) species with a *linear* relation

$$\rho = \rho_R + \frac{\partial \rho}{\partial c} (c - c_{\rho_R}), \quad \left\{ \rho_{ref}, \frac{\partial \rho}{\partial c}, c_{\rho_R} \right\} \tag{25}$$

Item 4c) then reads the parameter for the linear relations

$$\left\{ k_i, \frac{\partial \rho}{\partial c_k}, c_{k,\rho_R} \right\}_{i=1...NSRhoEOS} \tag{26}$$

where

i = the number in the list $1...NSRhoEOS$

k_i = the MT3DMS species number for this relation

c_k = the concentration of this species

c_{k,ρ_R} = the concentration of this species when the water has its reference density

Note that the reference density, DENSEREF, itself is the same for all species and read in separately in item 4a). This is done together with parameters that specify the relation between density and pressure head $\frac{\partial \rho}{\partial \phi_p} \simeq 4.46 \times 10^{-3}\,\mathrm{kg/m^4}$ in terms of the reference density:

$$\Delta \rho_P = \frac{\partial \rho}{\phi_p} (\phi_p - \phi_{p_R})$$

Clearly, NSRhoEOS>=0, otherwise no species are available to compute the density.

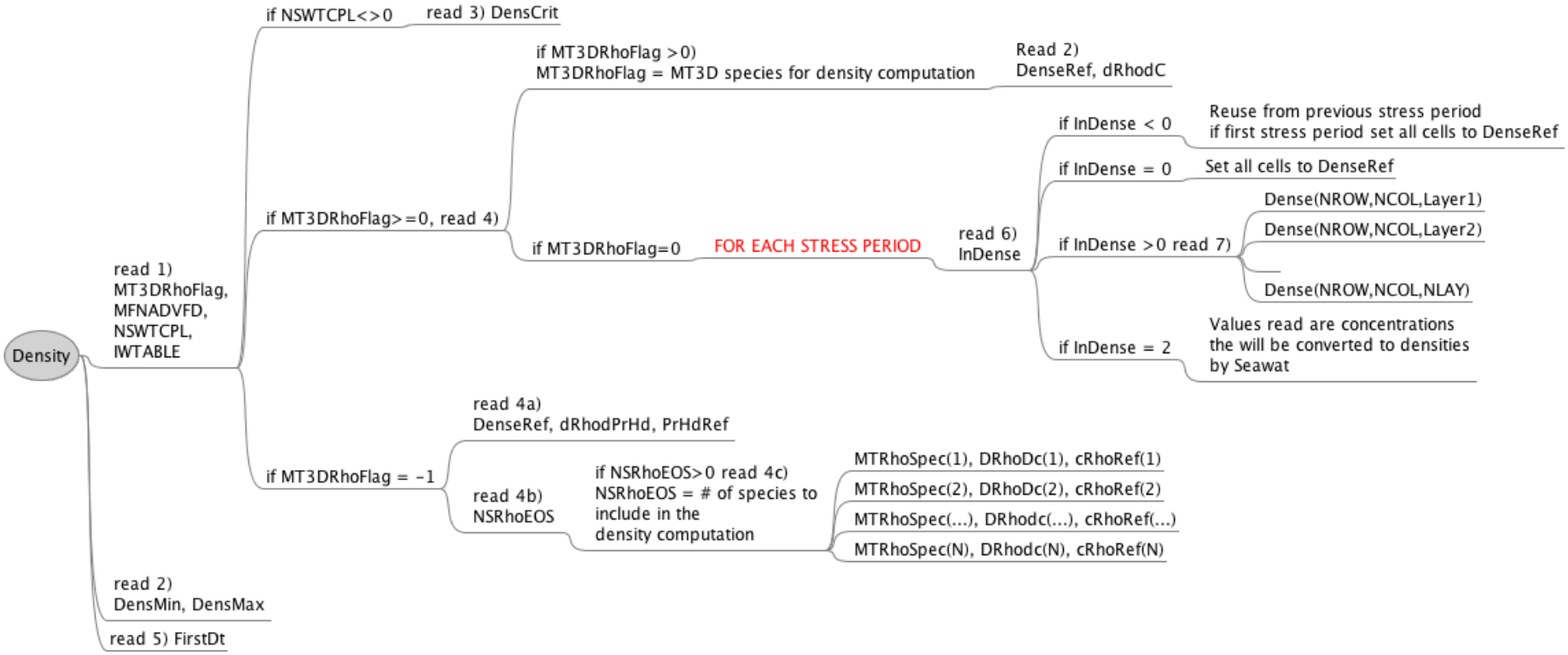

Figure 14: Density input scheme SEAWAT V4

17 Viscosity package

Figure 15 shows a mindmap of the input instructions of the Seawat V4 manual.

17.0.2 MT3DMUFLAG ($\mu Flag$)

$\mu Flag$is the major 3-way switch in the viscosity package. It can be -1, or >-1 (i.e. $\geqq 0$).

$\mu Flag \geqq 0$ If $\mu Flag \geqq 0$ then then only 1 MT3DMS species is involved in the viscosity computation in a simple linear relation. And it is obliged to specify for this species the three parameters needed for a linear computation of the relation between viscosity and this species' concentration

$$\mu = \mu_{ref} + \frac{\partial \mu}{\partial c}\left(c - c_{\mu_{Ref}}\right), \quad \left\{\mu_{ref},\, \frac{\partial \mu}{\partial c},\, c_{\mu_{Ref}}\right\} \tag{27}$$

This shows that any species can be used in this way to compute viscosity linearly, including but not necessarily, temperature.

The manual says that $\mu flag$ is the MT3DMS species number, but this conflicts with$\mu Flag = 0$, being an illegal species number.

if $\mu Flag > 0$, then $\mu Flag$ is the MT3DMS species number used for the concentration in (25).

$\mu Flag = 0$ if $\mu Flag = 0$, then viscosity will be read in for each stress period, but only if $INVISC > 0$ (item 4) for that stress period.

This implies that we can still have stress periods with $INVISC = 0$ and at the same time $\mu Flag = 0$, so that then Seawat may only compute viscosity using the parameter specified by (25) in item 3, without a species number being specified. From a user's perspective it is unclear how Seawat does this, , without involving any MT3DMS species or using some MT3DMS default species.

Directly reading of cell-viscosity values will be seldom. It's most likely application is a restart from a previous run. In that case, one may as well read in temperature or related species directly instead of viscosity.

- Items that are needed per stress period are specified in *mfLab* in the PER worksheet column "INVISC" of the workbook for the problem on hand. If INVISC is 1 for a stress period, then *mfLab* expects to find the specification of the viscosities to be read in the workspace parameter VISC which must be a cell array with the cell corresponding to the stress period for which INVISC==1 holding the 3D array with viscosity values for all cells of the model.

$\mu Flag < 0$, ($\mu Flag = -1$) is probable the only setting that you will ever use. It allows to include the concentration of zero or more species in the viscosity equation in a simple linear way, but additionally allows to use a sophisticated equation that relates temperature to viscosity.

If $\mu Flag < 0$, Seawat will compute viscosity using NSMUEOS (zero or more) species with a *linear* relation and, optionally and additionally to NSMUEOS, by a non-linear relation between temperature and viscosity.

Item 3d then reads the parameter for the linear relations

$$\left\{k_i,\, \frac{\partial \mu}{\partial c_k},\, c_{k,\mu_{Ref}}\right\}_{i=1...NSMUEOS} \tag{28}$$

where

i = the number in the list $1...NSMUEOS$

k_i = the MT3DMS species number for this relation

c_k = the concentration of this species

$c_{k,\mu_{Ref}}$ = the concentration of this species when the water has its reference viscosity

Note that the reference viscosity, VISCREF, itself is the same for all species and read in separately in item 3a).

Clearly, temperature may be one of the species just specified, but then it can only have a linear relation with viscosity. This is not generally sufficient. Therefore, the NSMUEOS species for linear relations are most suitable for the relation between viscosity and the concentration of certain species that affect it measurably.

It is also clear that NSMUEOS=0 is acceptable, as it means that no species affects viscosity in a linear fashion.

The relation between temperature and viscosity is specified using the MUTEMTOPT flag that is read in together with NSMUEOS.

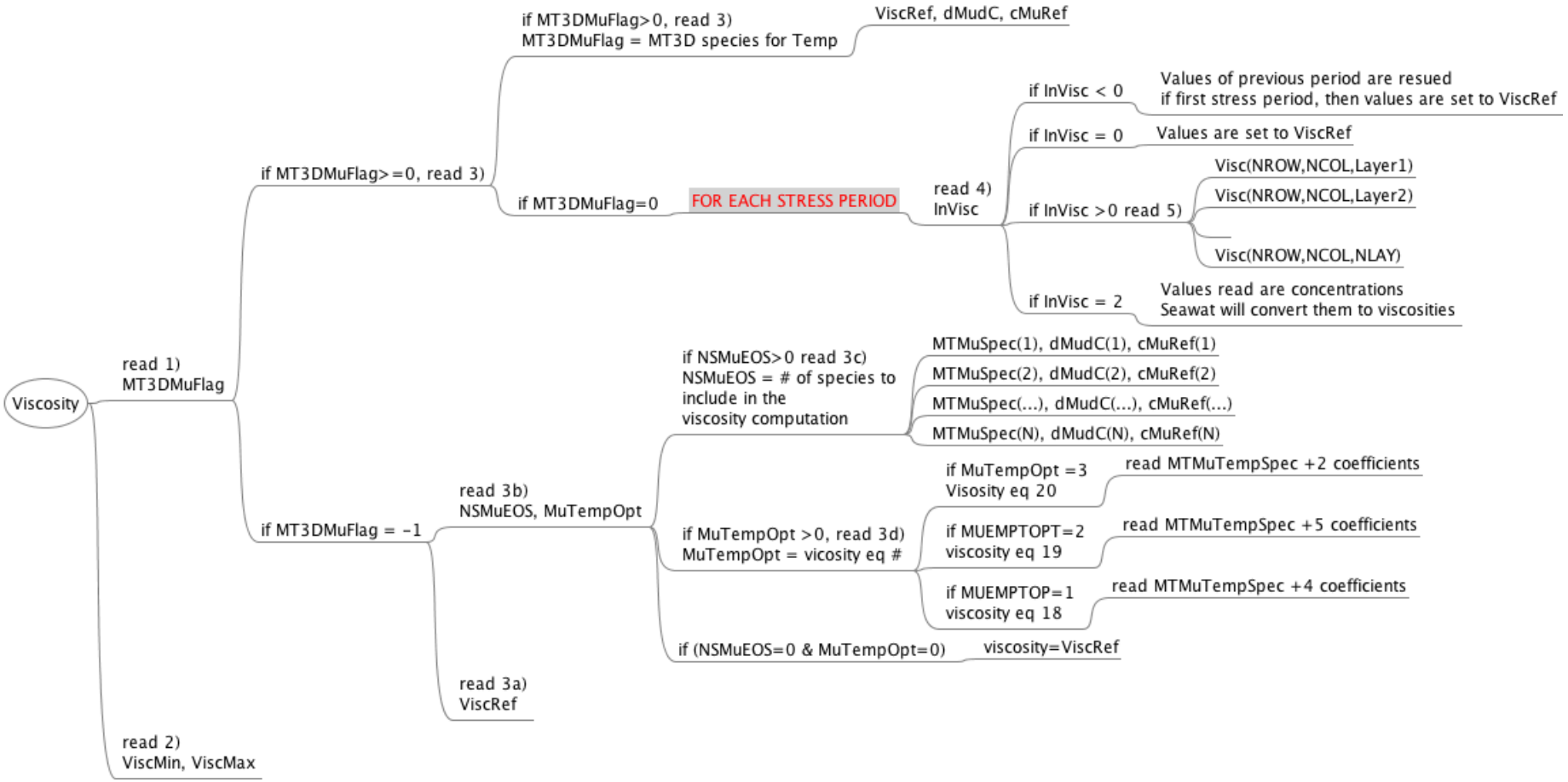

Figure 15: Viscosity input scheme SEAWAT V4

17.0.3 MUTEMPOPT (μ temperature option)

MUTEMPOPT is read in together with NSMUEOS in item 3b). MUTEMPOPT can be 0, 1, 2 or 3. If it is 0 and NSMUEOS=0 then the viscosity is fixed to VISCREF=μ_{Ref} in the entire model. If it is 1, 2 or 3 Seawat will compute the viscosity using a non-linear relation with temperature, specified in equation 18, 19 en 20 of the manual, on page 6.

Each of these equations has its own set of parameters (2, 5, and 4 respectively), which has to be specified in the input, headed by the MT3DMS species that is used for the temperature. This is done in item 3).

Note that this non-linear temperature relation is specified completely separated from the species involved in NSMUEOS. Therefore, the species number MTMUTEMPSPEC (see 26) must be different from any of the species numbers specified under NSMUEOS in item 3c) and it must be the species holding the temperature.

18 Axially Symmetric Modeling in MODFLOW, MT3D or SEAWAT

More ofthen than not, an axially symmetric model, similar to a cross section is very useful. In mfLab, setting the swith AXIAL=1 in mf_adapt turns the model in an axially-symmetric model. Setting AXIAL=0 is flat mode, which is the default. Axially symmetric model is implemented by multiplying certain model arrays by $2\pi\,|x_m|$ with x_m the x-coordinate of the cell centers. Multiplication is carried out in mf_setup, just before turing the arrays over to their writing routines that generate the input files for the different programs. Therefore, this process is out of view from the user, who only has to manage the switch in mf_adapt by including a line AXIAL=1.

This procedure has several consequences. For example, a model with running from a negative value to a positive one, actually implements two overlapping axially symmetric models, because of the multiplication with $|x_m|$. This feature can be used to obtain a symmetric figure showing the well in the center. It can also be used to simulate multiple axially symmetric cross sections at the same time, by setting the horizontal anisotropy to zero, so that the rows of the models are mutually disconnected. Also be aware that input of wells have dimension $\left[L^3/t\right]$ and no longer $\left|L^2/t\right|$ as is the case with flat models (of 1 m width). Each well in an axially symmetric model is in fact a circular extraction of injection ring of radius $|x_m|$. Also if you use two half axially symmetric models by applying an x-axis from negative to positive coordinates both sides are multiplied by $2\pi\,|x_m|$. The same is true for recharge and evapotranspiration. To make sure the reaction package works also the distribution coefficient is multiplied by $2\pi\,|x_m|$.

It is quite straightforward to do so and the examples have some. See for instance the FFSErad dirctory under examples/swt_v4/FSSE or the Goetherm2 directory under examples/swt_v4/Diffusion and heat/Geotherm2. The best example will be in mfLab/examples/swt_v4/CheckAxiBalance. This example not implements a mass transport and temperature, transport and proper visualisation and animation. It allows setting the switch AXIAL to 1 or 0 as to switch between flat and axial-symmetric mode. It finally shows in a graph the relation between the total mass injected and the total mass in the model as a function of time. Both should perfectly match in both the flat and axially symmetric model of the model, which they do.

18.1 The flow model

Modflow solves the following equation

$$\frac{\partial}{\partial x}\left(k_x\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial z}\left(k_z\frac{\partial\phi}{\partial z}\right)+q=S_s\frac{\partial\phi}{\partial t}$$

In axial-symmetric coordinates and consiering rings of length $2\pi r$, this becomes

$$\frac{\partial}{\partial r}\left(2\pi rk_r\frac{\partial\phi}{\partial r}\right)+\frac{\partial}{\partial z}\left(2\pi rk_z\frac{\partial\phi}{\partial z}\right)+2\pi rq=2\pi rS_s\frac{\partial\phi}{\partial t}$$

This equation immediately makes clear that we can approximate axial-symmetric flow by multiplying the horizontal and vertical conductivities and the storage coefficient by $2\pi r$.

We discretize by dividing r into small pieces dr wide (where dr may vary). We also divide the z-axis into layers of varying thickness dz. Then considering one such ring of size $2\pi r\Delta r\Delta z$, the water balance equation for this ring becomes:

$$\int\int\frac{\partial}{\partial r}\left(2\pi rk_r\frac{\partial\phi}{\partial r}\right)drdz+\int\int\frac{\partial}{\partial z}\left(2\pi rk_z\frac{\partial\phi}{\partial z}\right)drdz+2\pi\int\int rqdrdz=2\pi\int\int rS_s\frac{\partial\phi}{\partial t}drdz$$

Which is approximately assuming constant material properties within the ring and vertical components within the ring independent of r:

$$\Delta z \int \frac{\partial}{\partial r}\left(2\pi r k_r \frac{\partial \phi}{\partial r}\right) dr + \Delta r \int \frac{\partial}{\partial z}\left(2\pi r k_z \frac{\partial \phi}{\partial z}\right) dz + Q = 2\pi r \Delta r \Delta z S_s \frac{\partial \phi}{\partial t}$$

where $\int 2\pi r dr \approx 2\pi r \Delta r$. But of course, it would be better to use the exact value of $\pi\left(r_{i+1}^2 - r_i^2\right)$. Further, Q is the inflow for the entire ring. When making the switch to axial-symmetric flow we may ignore the vertical compoments as they are not altered except for the multiplication by $2\pi r$ which is applied by multiplying the conductivities by this factor. The integration of this first term yields

$$Q_r|_{r+\frac{\Delta r}{2}} - Q_r|_{r-\frac{\Delta r}{2}} = 2\pi \Delta z \left(r + \frac{\Delta r}{2}\right) k_r \frac{\partial \phi}{\partial r}|_{r+\frac{\Delta r}{2}} - 2\pi \Delta z \left(r - \frac{\Delta r}{2}\right) k_r \frac{\partial \phi}{\partial r}|_{r-\frac{\Delta r}{2}}$$

Which shows that simpleminded multiplication of k_r by $2\pi r$ is inacurate when not $r + \frac{\Delta r}{2} \approx r - \frac{\Delta r}{2}$ with r the radius of the center of the ring and Δr its width.

We may obtain a more accurate solution by computing the head difference between adjacent rings by assuming the horizontal flux between the center of adjacent rings constant and integrating the head gradient between these centers:

$$\Delta \phi_{r_i \to r_{i+1}} = \frac{Q_{r_i + \frac{\Delta r_i}{2}}}{2\pi \Delta z}\left\{\frac{1}{k_r} \ln\left(\frac{r_i + \frac{\Delta r_i}{2}}{r_i}\right) + \frac{1}{k_{r_{i+1}}} \ln\left(\frac{r_{i+1}}{r_{i+1} - \frac{\Delta r_{i+1}}{3}}\right)\right\}$$

So that

$$Q_{r_i + \frac{\Delta r_i}{2}} = \frac{2\pi \Delta z}{\frac{1}{k_{r_i}} \ln\left(\frac{r_i + \frac{\Delta r_i}{2}}{r_i}\right) + \frac{1}{k_{r_{i+1}}} \ln\left(\frac{r_{i+1}}{r_{i+1} - \frac{\Delta r_{i+1}}{3}}\right)} \left(\phi_{r_{i+1}} - \phi_{r_i}\right)$$

and

$$Q_{r_i - \frac{\Delta r_i}{2}} = \frac{2\pi \Delta z}{\frac{1}{k_{r_{i-1}}} \ln\left(\frac{r_{i-1} + \frac{\Delta r_{i-1}}{2}}{r_{i-1}}\right) + \frac{1}{k_{r_i}} \ln\left(\frac{r_i}{r_i - \frac{\Delta r_i}{3}}\right)} \left(\phi_{r_i} - \phi_{r_{i-1}}\right)$$

If the adjacent cells happen to have the same conductivity, then

$$Q_{r_i + \frac{\Delta r_i}{2}} = \frac{2\pi k_r \Delta z}{\ln\left(\frac{r_{i+1}}{r_i}\right)} \left(\phi_{r_{i+1}} - \phi_{r_i}\right)$$

$$Q_{r_i - \frac{\Delta r_i}{2}} = \frac{2\pi k_r \Delta z}{\ln\left(\frac{r_i}{r_{i-1}}\right)} \left(\phi_{r_i} - \phi_{r_{i-1}}\right)$$

If MODLFOW would compute the horizontal compoments in this way, the model would be very accuarate even if the width of the rings was substantial relative to the radius itself. Thisis not the case. However, if the transmissivity is a linear function of the radius, which is the case here if the conductivifty of the adjacent cells in the same and we multiply these values by $2\pi r$, then MODFLOW can yield exact results for radial flow:

The idea is in fact to compute the flow between adjacent cell centers. In the case of linear transmissiviy change as is the case in the r-direciton if the conductivityis linear and $2\pi rk$ is used, with constant k

we have

$$\begin{aligned} d\phi &= \frac{Q}{T} dr \\ \Delta\phi &= \frac{Q}{\overline{T}} \Delta r \end{aligned}$$

Therefore,

$$\Delta\phi = \int d\phi = \frac{Q}{\overline{T}} \Delta r = \int \frac{Q}{T} dr$$

and so

$$\begin{aligned}\frac{1}{\overline{T}} &= \frac{1}{\Delta r}\int\frac{dr}{T}\\ &= \frac{1}{2\pi k\Delta z\,(r_2-r_1)}\int_{r_1}^{r_2}\frac{dr}{r}\\ &= \frac{1}{T_2-T_1}\ln\left(\frac{T_2}{T_1}\right)\end{aligned}$$

so that

$$\overline{T} = \frac{T_2-T_1}{\ln\left(\frac{T_2}{T_1}\right)} \tag{29}$$

The manual of MODFLOW2000 mentions this solution from Goode and Appel, which is implemented in MODFLOW by means of the LAYAVG. If the value is set to 1 the interblock conductivities are computed as in equation 29. This implies that the solution will be exact in the axial-symmetric case when the conductivitie of adjacent cells are the same. This is often the case. Therefore, mfLab makes sure the LAYAVG switch is set to 1 in the case AXIAL is set to 1.

The transmissivity between adjacent cells can further be adapted to the satuated thicknesses. In that case the LAYAVG flag should be set to 2.When axial-symmetric flow is simulated, *mfLab* will set the LAYAVG flag to 1 if it is zero and leave the user-set value in case it is not zero.

18.2 The transport model (with linear sorption)

The transport model MT3DMS and, therfore, SEAWAT can also be used in axial-symmetric mode by multiplying the appropriate variables by $2\pi r$.

The partial differential equation 30 for the transport of mass (concentration c) and 36 for heat (through using temperature) respectively are (also see the MT3DMS manual page 4):

$$\epsilon R\frac{\partial c}{\partial t} = \nabla\cdot(\epsilon D\nabla c) - \overrightarrow{q}\nabla c + \iota + Nc_N - Ec_E - \gamma\epsilon cR \tag{30}$$

In the equation 30, c is dissolved constituent concentration. R is retardation or better the total mass per unit bulk volume over the dissolved mass in the fluid in the case of linear equilibrium sorption. ϵ is effective porosity, $D\left[\mathrm{L}^2/\mathrm{T}\right]$ is molecular diffusion + Hydrodynamic dispersion, $\overrightarrow{q}\left[\mathrm{L}/\mathrm{T}\right]$ is advection, $\iota\left[\mathrm{M}/\mathrm{L}^3/\mathrm{T}\right]$ is a mass source term, i.e. total mass per unit volume and time, $Nc_N\left[\mathrm{T}^{-1}\mathrm{M}/\mathrm{L}^3\right]$ is mass provided by recharge $N\left[\mathrm{T}^{-1}\right]$ and Ec_E mass extracted by evaporation $E\left[\mathrm{T}^{-1}\right]$ both with their specific user-provided concentrations c_N and c_E. Note that equation 30 expresses source and sinks still as volume per unit voluem per time. After discretization of the equation, the dimension of N and E will become $[L/T]$ it will become Finally, $\gamma\epsilon cR = \gamma m\left[\mathrm{M}/\mathrm{L}^3/\mathrm{T}\right]$ is the breakdown (called irreversible reaction in the MT3DMS manual), if it occurs.

In linear sorption $R\,[-]$ is expressed using the distribution coefficient $K_d\left[\mathrm{L}^3/\mathrm{M}\right]$

$$R = 1 + \frac{\rho_b K_d}{\epsilon} = 1 + \frac{2\pi r\rho_b K_d}{2\pi r\epsilon} \tag{31}$$

where the second expression is the numerically equivalent form used in the axially symmetric case (see below), in which we multiply both ϵ and ρ_b by $2\pi r$.

In the axially symmetric situations, we consider mass transport for a ring of length $2\pi r$ with r the distance to the center of the model (33).

The transport equation is thus changed as follows by multiplying all terms by $2\pi r$:

$$(2\pi r\epsilon)\,R\frac{\partial c}{\partial t} = \nabla\cdot((2\pi r\epsilon)\,D\nabla c) - (2\pi r\,\overrightarrow{q})\,\nabla c + 2\pi r\iota + 2\pi rNc_N - 2\pi rEc_E - \gamma\,(2\pi r\epsilon c)\,R \tag{32}$$

or, equivalently

$$(2\pi r\epsilon)\frac{\partial c}{\partial t} = \nabla\cdot\left((2\pi r\epsilon)\frac{D}{R}\nabla c\right) - \frac{(2\pi r\overrightarrow{q})}{R}\nabla c + \frac{2\pi r\iota}{R} + \frac{2\pi rN}{R}c_N - \frac{2\pi rE}{R}c_E - \gamma\,(2\pi r\epsilon c) \tag{33}$$

We can achieve this in the model by multiplying ϵ and ρ_b both by $2\pi r$. The multiplication of ρ_b by $2\pi r$ follows from equation 31. We will also multiply recharge and evaporation by $2\pi r$ but assume other source terms as given by the user for the entire rings, so that nu multiplication is invoked for these terms.

18.3 Heat transport

The linear sorption considered in the transport equation above are essential for the heat transprot equation because heat exchange between the fluid and the solids obey the same mathematical rules.

At the risk of creating some confusion at this stage, we may introduce the important application of using the transprot model to simulate heat transport. This is done by converting the mass transport equation to the heat transport equations where heat replaces mass and temperature replaces concentration.

$$\rho c\frac{\partial T}{\partial t} = \nabla\cdot(\lambda\nabla T) - \epsilon\rho_w c_w\,\overrightarrow{v}\nabla T + e \tag{34}$$

In equation 36 $T\,[\mathrm{K}]$ is temperature and t is time. $\rho c\left[\mathrm{E/L^3/K}\right]$ is the total volumetric heat capacity of the medium, which includes both water and solids with bulk density $\rho\left[\mathrm{M/L^3}\right]$ and bulk heat capacity $c\,[\mathrm{E/M}]$.

$$\rho c = \epsilon\rho_w c_w + (1-\epsilon)\,\rho_s c_s$$

which is a numeric value and in which ϵ and $1-\epsilon$ are weights.

The bulk heat conductivity $\lambda\,[E/T/L/K]$ i.e. [W/m2/(/k/m)]=[W/k/m] can be computed as

$$\lambda = \lambda_w\epsilon + \lambda_s\,(1-\epsilon)$$

which is also a numerical value with ϵ and $1-\epsilon$ functioning as weights.

The last term, $e\left[\mathrm{E/L^3/T}\right]$ is a heat source or sink term depending on its sign. It may be split in different contributions as necessary for the problem to solve.

Clearly, but perhaps a bit confusing is that $c\left[\mathrm{M/L^3}\right]$ in equation 30 is the concentration of the dissolved constituent, which has nothing to do with $c\,[\mathrm{E/M/K}]$ or $c_w\,[\mathrm{E/M/K}]$ in equation 36, denoting the bulk and water mass heat capacity respectively.

As the retardation $R = \frac{m}{\epsilon c}$ in mass transport equals the ratio of the total bulk mass $m\left[\mathrm{M/L^3}\right]$ of the consitutuent, i.e. dissolved + sorbed mass, $\epsilon c\left[\mathrm{M/L^3}\right]$ the dissolved mass. This is the case of linear sorption. In the case of heat transport, i.e. conduction and convection, the retardation takes the form $R = \frac{\rho c}{\epsilon\rho_w c_w}$. Hence dividing equation 36 by $\rho_w c_w$ yields.

$$\epsilon\frac{\rho c}{\epsilon\rho_w c_w}\frac{\partial T}{\partial t} = \nabla\cdot\left(\epsilon\frac{\lambda}{\epsilon\rho_w c_w}\nabla T\right) - \overrightarrow{q}\nabla T + \epsilon\frac{\rho c}{\epsilon\rho_w c_w}\frac{e}{\rho c} \tag{35}$$

Which shows that the transport model will compute heat convection if apply linar sorption and set

$$R = \frac{\rho c}{\epsilon\rho_w c_w} = \frac{2\pi r\rho c}{2\pi r\epsilon\rho_w c_w},\;\; D = \frac{\lambda}{\rho_w c_w},\;\; \iota = \frac{e}{\rho c}$$

So that the heat transport equation becomes

$$\epsilon R\frac{\partial T}{\partial t} = \nabla\cdot\left(\epsilon\frac{\lambda}{\epsilon\rho_w c_w}\nabla T\right) - \overrightarrow{q}\nabla T + \epsilon R\frac{e}{\rho c} \tag{36}$$

This equation 36 is equivalent to the transport equation 30.

Considering rings in axial-symmetric flow, we multiply the heat capacity by $2\pi r$ to acknowledge the total heat capacity within the considered ring:

$$2\pi r\epsilon R\frac{\partial T}{\partial t} = \nabla\cdot\left(2\pi r\epsilon\frac{\lambda}{\epsilon\rho_w c_w}\nabla T\right) - (2\pi r\overrightarrow{q})\,\nabla T + 2\pi r\epsilon R\frac{e}{\rho c} \tag{37}$$

Dividing by R

As we have seen, the retardation remains unaltered in the axial-symmetric case by multiplying also ρ (the bulk density of the medium) by $2\pi r$. As before, we need not worry over the second (advection) term to ther

right, as $2\pi r\overrightarrow{q}$ is given by the flow model. Further, the right most term can be considered as the total source of heat (or sink) for a ring to be provided by the user. Finally, we see that we have to replace the diffusion coefficient in the mass transport model, D by

$$D = \frac{\lambda}{\epsilon\rho_w c_w} = \frac{2\pi r\lambda}{2\pi r\epsilon\rho_w c_w} \tag{38}$$

The second form of equation 38 is numerically equivalent to its first from, but we have to write it that way because ϵ will be changed into $2\pi r\epsilon$ when switching to axial-symmetric flow. This implies that if we do not precompute D using the first from, we have to multipy λ by $2\pi r$ as well in the case of axial-symmetric flow. However, MT3DMS and SEAWAT require D to be specified, so tha it is always precomputed using the first form.

Dividing equation 39 by retardation R and using equation 38 yields

$$2\pi r\epsilon\frac{\partial T}{\partial t} = \nabla\cdot\left(2\pi r\epsilon\frac{D}{R}\nabla T\right) - \frac{(2\pi r\overrightarrow{q})}{R}\nabla T + 2\pi r\epsilon\frac{e}{\rho c} \tag{39}$$

which is equivalent to equation 33. Note that the source term factor $\frac{e}{\rho c}$ has dimension termpeature per unit time caused by the heat injection of e.

18.4 Sorption

As was seen above, the essential things is that the retardation must nut change when switching to axial-symmetric flow. Retardation is given for linear sorptioin by equation 31. In fact is is the ratio of total mass over dissolved mass or energy per unit volume:

$$R \cong \frac{m}{\epsilon c} \cong \frac{\rho c}{\epsilon\rho_w c_w} \tag{40}$$

With linear sorption the retardation must be obtatained through the MT3DMS parameters ρ_b, K_d and ϵ:

$$R = 1 + \frac{\rho_b K_d}{\epsilon}$$

Of course we can precompute K_d starting from a desired or given retardation as

$$K_d = \frac{\epsilon}{\rho_b}(R-1) \tag{41}$$

In the case of heat transport we have

$$K_d = \frac{\epsilon}{\rho_b}\left(\frac{\rho_b c_b}{\epsilon\rho_w c_w} - 1\right) \tag{42}$$

$$= \frac{c_b/c_w}{\rho_w} - \frac{\epsilon}{\rho_b} \tag{43}$$

Equation 42 is as opaque as is 43, equation 41 is general and convenient, while R may also be taken form equation 40.and it can be This is automatically achieved in the axial-symmetric case by multiplying both ρ_b and ϵ by $2\pi r$ and leaving K_d unaltered.

18.5 What to do with the reaction coefficients SP1 and SP2 in case of axial-symmetric flow

In MT3DMS the sorption otions and chemical reaction options are set using the parameters ISOTHM, SP1 and SP2. The ISOTHM parameter is use as follows:

ISOTHM=0 % no sorption.
ISOTHM=1 % linear sorprion, equilibrium controlled.
ISOTHM=2 % Freundlich isotherm, equilibrium controlled.
ISOTHM=3 % Langmuir sorption, equilibrium controlled.
ISOTHM=4 % first-order kinetic sorption, non-equilibrium.
ISOTHM=5 % dual-domain mass transfer without sorption.
ISOTHM=6 % dual-domain mass transfer with sorption.

Depending on the value of ISOTHM, the first and second reaction coefficients SP1 and SP2 assume a different meaning.

We will consider how to handle the coefficients SP1 and SP2 used in the reaction package of MT3DMS in the case of axial-symmetric flow, that is when AXIAL=1 is set in *mf_ adapt.m*. The idea is that the user should not have to worry about this at all, he or she only has to set the AXIAL parameter somewhere in *mf_ adapt.m*. Clearly leaving out AXIAL altogther or setting AXIAL=0 implies that the cross section is flat.

Sorption, whether linear or not can be expressed by equation 14 on page 12 in the manual of MT3DMS giving equation 44 for non-linear retardation.

$$R = 1 + \frac{\rho_b}{\epsilon}\frac{\partial \overline{c}}{\partial c} \tag{44}$$

where $\overline{c}\,[\mathrm{M_{consituent}/M_{solids}}]$ is the sorbed concentration and $c\,[\mathrm{M_{consituent}/Volume_{water}}]$ the dissolved concentration. As we have seen above, the numerical value of the retardation is unchanged by the switch from linear to axial symmetric flow. Becuase we multiply both ρ_b and ϵ by $2\pi r$ in the case of axial-symmetric flow, R remains unchanged as long as $\partial\overrightarrow{c}/\partial$c is he same in both situatons.

18.5.1 ISOTHM=0, SP1 and SP2 are not used

Nothing to comment here.

18.5.2 ISOTHM=1, Linear sorption ($SP1 = K_d$ and $SP2$ is not used)

With linear sorption we have

$$\frac{\partial \overline{c}}{\partial c} = K_d\left[\frac{\mathrm{M_{constituent}/M_{solids}}}{\mathrm{M_{consitutent}/Volume_{warer}}}\right] = \left[\frac{\mathrm{Volume_{water}}}{\mathrm{M_{solids}}}\right] = \left[\mathrm{L^3/M}\right]$$

In the case of linear sorption SP1=K_d and SP2 is read but not used. As we have seen above, K_d remains untouched by the switch to axial-symmetric flow.

18.5.3 ISOTHM=2, Non-linear sorption, Freundlich Isotherm

In the case of Freundlich isotherms we have

$$\frac{\partial \overline{c}}{\partial c} = aK_f c^{a-1}$$

SP1=K_f and SP2=a, the Freundlich exponent. Both coeffiicents remain unaltered.

18.5.4 ISOTHM=3, Non-linear sorption, Langmuir

In the case of Langmuir sorption

$$\frac{\partial \overline{c}}{\partial c} = \frac{K_l \overline{S}}{\left(1 + K_l c\right)^2}$$

SP1=K_l and SP2=$\overline{S}\,[\mathrm{M/M}]$ is the total concentration of the surface asortion places available.
Both SP1 and SP2 remain unaltered.

18.5.5 ISOTHM=4, non-equilibrium sorption

SP1=K_d, SP2= β, the first order mass transfer rate between the dissolved and sorbed phases $\left[T^{-1}\right]$.

$$\rho_b \frac{\partial \overline{c}}{\partial t} = \beta\left(c - \frac{\overline{c}}{K_d}\right)$$

This can be transformed into the form of the retardation equation as follows:

$$\frac{\partial \overline{c}}{\partial c} = \frac{\beta}{\rho_b \frac{\partial c}{\partial t}}\left(c - \frac{\overline{c}}{K_d}\right)$$

SP1=K_d and SP2=β. These coefficients remain untouched when AXIAL=1.

18.5.6 ISOTHM=5, dual medium mass transfer without sorption

SP1=read but not used, SP2=κ, the first order mass transfer rate between the two domains $\left[T^{-1}\right]$.
SP1 and SP2 are unaltered in the case of axial-symmetric flow.

18.5.7 ISOTHM=6, dual medium mass transfer with sorption

SP1=K_d, SP2=β, the first order mass transfer rate between the two domains $\left[T^{-1}\right]$.
SP1 and SP2 are unaltred by a switch to axial-symmetric flow.

18.6 Reaction rates

Reaction rates may be specified for each mobile species RC1 and for each immobile species separately. The equations on page 15 and 16 of the MT3DMS manual talk about the first order reaction rates of the liquid phase and the first order reaction rates of the sorbed phase. Both can be different for the mobile portion and the immobile portion (liquid in dead-end pores and sorbed onto moving particles). These reaction rates are not multiplied by $2\pi r$ in the case of axial-symmetric flow as they are not dependent on the grid or space, they are just relative to the current concentration.

18.7 Example

Figure 16 was made in the example mfLab/examples/swt_v4/Diffusion and heat/Geotherm2. With this information it is now straightforward to make any radial-symmetric model, cross section model or full 3D model. The examples is an axial symmetric flow case showing injection of a cool fluid into a geothermal aquifer at great depth after 50 years of injection. The layer is marked by the thin green line which denots the position of the injected fluid after 50 years of injection, while the colors denote tempeatures between 30 and 70 degrees C. The temperature is delayed relative to the water due to heat exchange between the water and the grains. The temperature also penetrates the overlying and underlying layers while the water does not due to their low conductivity, which does not hamper temperature presentation. Applied parameters are in the caption of the figure.

References

[Langevin (2008)] Langevin C.D. (2008) Modeling Axisymmetric Flow and Transport. Ground Water. Vol. 46 (4), pp 579-590.

[Savage (2006)] Savage, C. (2006) http://cfis.savagexi.com/2006/05/03/google-maps-deconstructed

[Marcator (2011)] Wikipedia (2011) Mercator Projection. http://en.wikipedia.org/wiki/Mercator_projection

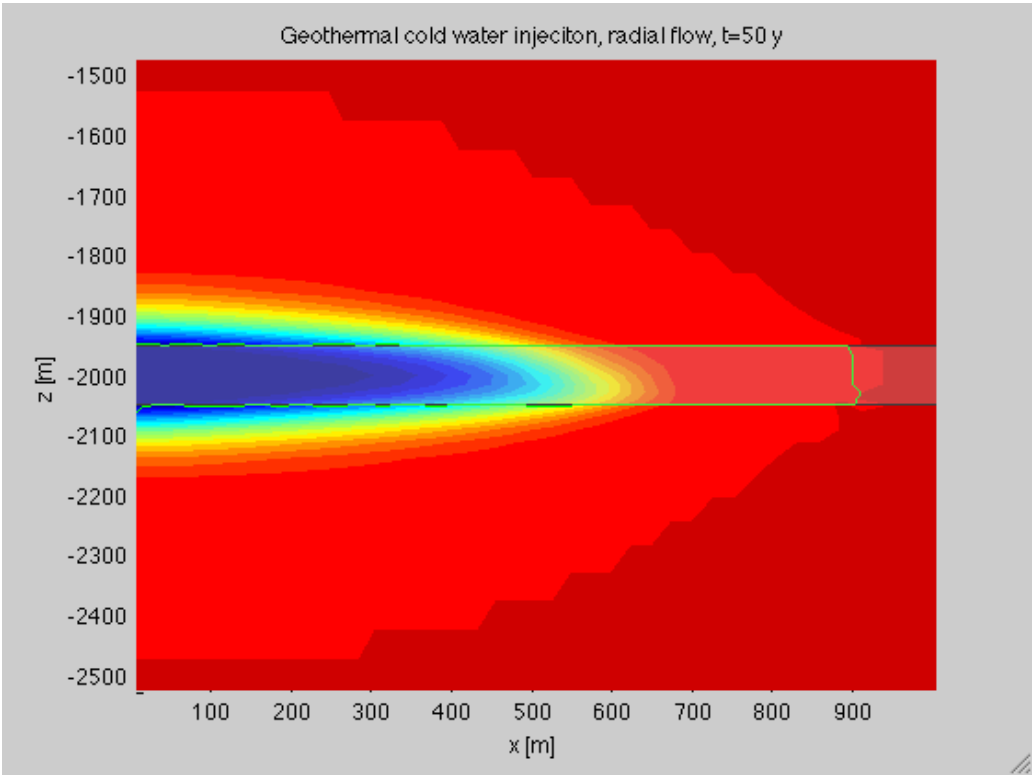

Figure 16: Temperature after 50 years of injection of 20C water in a 70C environment. The flow is axially symmertric. Viscosity and temperature-dependent density are taken into account. The thin green line is the position of the water front. Aquifer $D = 100\,\mathrm{m}$, $k = 1\,\mathrm{m/d}$, $\epsilon = 0.2$, $Q_{inj} = 200\,\mathrm{m^3/h}$, $\lambda_w = 0.06\,\mathrm{W/m/K}$, $\lambda_s = 3\,\mathrm{W/m/L}$, $\lambda = 2.412\,\mathrm{W/m/K}$, $\rho_w = 1000\,\mathrm{kg/m^3}$, $\rho_s = 2650\,\mathrm{kg/m^3}$, $\rho_b = 2320\,\mathrm{kg/m^3}$, $c_w = 4200\,\mathrm{J/kg/K}$, $c_s = 800\,\mathrm{J/kg/K}$, $\rho_w c_s = 4.2e+06\,\mathrm{J/m^3/K}$, $\rho_w c_s = 2.12e+06\,\mathrm{J/m^3/K}$, $\rho_c = 2.536e+06\,\mathrm{J/m^3/K}$, $K_{dtemp} = 1.74e-4\,\mathrm{m^3/kg}$, $D_{temp} = 50\,\mathrm{K}$ – temperature drop in geothermal system. Computaion method MOC.