TTim
A Multi-Aquifer Transient Analytic Element Model
Version 0.3

DRAFT

Mark Bakker
Water Resources Section, Civil Engineering and Geosciences
Delft University of Technology, Delft, The Netherlands
mark.bakker@tudelft.nl

August 20, 2015

Contents

1 Introduction

TTim (pronounce ”Tee Tim”) is a solver for transient multi-layer flow based on the Laplace-transform analytic element method and is developed at the Delft University of Technology in Delft, The Netherlands.

2 Funding

Version 0.3 was funded by Intera, Inc., Austin, TX. Version 0.01-0.23 was funded by Layne Hydro, a division of the Layne Christensen Company, in Bloomington, IN. Version 0.01 of TTim was funded by the US EPA Ecosystems Research division on contract QT-RT-10-000812 to S.S. Papadopulos and Associates in Bethesda, MD. Version 0.01 was developed in collaboration with S.S. Papadopulos and Associates in Bethesda, MD, and Layne Hydro in Bloomington, IN.

3 License

4 TTim website

To be added.

5 Installation

The following free, open-source software is required to run TTim: Python version 2.X, the packages numpy, scipy, and matplotlib. The shapely package is needed for the use of Lake elements (optional). TTim uses a FORTRAN extension which means that TTim is unfortunately both platform dependent and Python version dependent.

Although it is possible to download all required Python packages separately, it is much easier to install one of the Python installers that comes with a large selection of popular packages. Three main options are Anaconda, Enthought, and PythonXY. This manual is written for Anaconda users. Use of the other distributions only differs in how IPython is started.

5.1 Running TTim on a platform other than Windows or Mac

The only reason TTim is platform dependent is that some functions are coded in FORTRAN and compiled as FORTRAN extensions to improve performance. Compiled versions are included in the Windows and Mac installers. The FORTRAN files `bessel.f95` and `invlap.f90` may be compiled to Python extensions on any other platform with f2py.

5.2 Differences with versions before 0.3

TTim version 0.3 is not compatible with previous versions. The most important change is that the numbering of layers starts with 0 (zero) for the top layer, where previous versions started at 1. The zero-based numbering is consistent with the numbering of indices in Python.

5.3 Release history

Verison 0.01. November 2010.
Version 0.02. October 2011.
Version 0.1. February 2012.
Version 0.11. April 2012.
Version 0.2. July 2012.
Version 0.23. September 2013.
Version 0.3. May 2015.

6 Main approximations

The theory behind TTim is described in detail in Bakker (2013a), while a more applied discussion, including a practical example, is published in Bakker (2013b). Benchmark problems for wells are published in Louwyck et al. (2012).

TTim is developed to simulate transient flow in an aquifer system consisting of an arbitrary number of layers. The term layers refers to both aquifer layers and leaky layers (also referred to as aquitard layers). Aquifer system properties are homogeneous within an aquifer (hydraulic conductivity, thickness, and storage), even when a phreatic surface is modeled in the top layer, and within a leaky layer (thickness, resistance, and specific storage). Both aquifer layers and leaky layers are numbered from the top of the aquifer system to the bottom, starting with leaky layer 0 on top of aquifer layer 0 (Yes, they start at zero, just like the index of arrays in numpy).

Vertical resistance to flow is neglected within an aquifer layer, in accordance with the Dupuit approximation. Flow in leaky layers is vertical. Heads are computed in aquifer layers only. The governing system of differential equations is presented in Hemker and Maas (1987). Principles of the Laplace-transform analytic element method are outlined in Kuhlman and Neuman (2009) and Bakker and Kuhlman (2011).

Boundary conditions specified for each element (for example, specified head or discharge) are zero for $t < 0$. Conditions in the aquifer at $t = 0$ are approximated as steady-state. TTim computes heads with respect to this steady-state situation. This means that the head along a stream with a constant water level needs to be specified as zero. Analytic elements may start, stop or change at any time and are constant for each specified period. Analytic element solutions are computed in the Laplace domain. A solution in the physical domain is obtained through numerical inversion using the algorithm of De Hoog et al. (1982). The aquifer domain is infinite with the head at infinity equal to zero in all aquifer layers.

Units are not specified in the model. The user must use consistent units when specifying aquifer properties and interpreting model results.

7 Types of elements and their practical application

Five kinds of elements are implemented: wells, line-sinks, line-doublets, and circular area-sinks. Most elements have a number of types.

7.1 Wells

For wells, the following types of elements are implemented:

- Wells that are screened in multiple layers for which only the total discharge is known. Command: `Well`. Practical application: The `Well` is used to simulate flow to a well that is screened in multiple layers; only the total discharge of the well is specified. TTim distributes the discharge across the screens such that the head inside the well is the same in all layers. Well bore storage and skin effect may be taken into account.

- Wells that are screened in multiple layers for which the total discharge is zero. Command: `ZeroMscreenWell`. Practical application: A `ZeroMscreenWell` represents an abandoned well or an observation well that is screened in multiple layers. If the abandoned well is filled up with material, a resistance to vertical flow may be specified, equal to the distance between the centers of two screens divided by the vertical hydraulic conductivity of the fill material. In case of an observation well that is screened in multiple layers, the vertical resistance is commonly zero.

- Wells with a specified head. The same head is used for each layer that the well is screened in. Command: `HeadWell`. Practical application: `HeadWell` elements may be used to compute the discharge of a well for a desired drawdown or injection head.

- Wells with a specified discharge. The discharge is specified and is the same for each layer that the well is screened in. Command: `DischargeWell`. Practical application: this element is used to simulate flow to a well that pumps with a known discharge. It is not common to use this element for a well that is screened in multiple layers, as the discharge is the same in each layer. Hence, when a discharge Q is specified, Q is taken from each layer that the well is screened in. This may be useful, however, when comparing to an exact solution for a well with a uniform inflow along the screen.

- Wells with a specified head equal to zero. Command: `ZeroHeadWell`. Practical application: `ZeroHeadWell` elements are not used much in practice.

All wells may, optionally, have an entry resistance. For wells the entry resistance represent the skin effect of the well. The discharge Q for a layer is computed as:

$$Q = 2\pi r_w H \frac{h_{\text{out}} - h_{\text{in}}}{c} \tag{1}$$

where r_w is the radius of the well, H is the aquifer thickness, c is the entry resistance (with units of time), and h_{out} and h_{in} are the head just outside and just inside the well, respectively. Kruseman and De Ridder (1990) and some others define the skin effect slightly differently. Their additional drawdown due to the skin effect s_s is defined as

$$s_s = \frac{Q}{2\pi T} c_s \tag{2}$$

where c_s [-] is their coefficient for the skin effect. For the case of a single well in an infinite aquifer, the skin effect parameter c_s may be expressed in terms of c as

$$c_s = \frac{k}{r_w} c \tag{3}$$

7.2 Line-sinks

Line-sinks are commonly entered as strings, which are discussed in the next section, but may also be entered as individual elements. For line-sinks, the following types of elements are implemented:

- Line-sinks with a specified head. The same head is used for each layer that the line-sink is screened in. Command: `HeadLineSink`. Practical application: `HeadLineSink` may be used to simulate flow to rivers and streams with a known and variable water level.
- Line-sinks with a specified head equal to zero. Command: `ZeroHeadLineSink`. Practical application: `ZeroHeadLineSink` may be used to simulate flow to rivers and streams with a fixed water level.
- Line-sinks that are screened in multiple layers for which the total discharge is zero. Command: `ZeroMscreenLineSink`. Practical application: A `ZeroMscreenLineSink` represents a vertical fault with a high vertical hydraulic conductivity. A vertical resistance to flow inside the fault and a width of the fault may be specified. The vertical resistance equals the distance between the centers of two adjacent layers divided by the vertical hydraulic conductivity of the fill material.
- Elements with specified discharge. The discharge is specified and is the same for each layer that the element is screened in. Command: `LineSink`. Practical application: The `LineSink` is not used much in practice, as the inflow is rarely known.
- Elements that are screened in multiple layers for which only the total discharge is known. Command: `McreenLineSink`. Practical application: The `MscreenLineSink` is not used much in practice, as the inflow is rarely known.

An entry resistance may be specified, optionally, for each line-sink type. The entry resistance may represent a clogged streambed or the resistance encountered by three-dimensional path lines, for example. The total discharge of a line-sink is computed as

$$Q = Lw \frac{h_{\text{out}} - h_{\text{in}}}{c} \tag{4}$$

where L is the strength of a line-sink, w is the distance over which water enters the line-sink, c is the entry resistance (with units of time), and h_{out} and h_{in} are the head just outside and just inside the line-sink, respectively. The distance w may be the width of the stream, for example, in case of a partially penetrating stream. In case the stream penetrates the aquifer layer fully, the distance w may equal the thickness of the aquifer layer (if water enters primarily from one side), or twice the aquifer thickness (if water enters from both sides). Note that the quantity that matters is really w/c.

7.3 Line-sink strings

The following line-sink string types have been implemented:

- A string of line-sinks with a specified head. Command: `HeadLineSinkString`. A string of `HeadLineSink` elements all with the same head variation.
- A string of line-sinks with a specified head equal to zero. Command: `ZeroHeadLineSinkString`. A string of `ZeroHeadLineSink` elements.
- A string of line-sinks that are screened in multiple layers for which the total discharge of each line-sink in the string is zero. Command: `ZeroMscreenLineSinkString`. A string of `ZeroMscreenLineSink` elements.
- A string of line-sinks that is screened in one or multiple layers for which the total discharge of *the entire* string is specified. Command: `MscreenLineSinkDitchString`. The discharge is distributed across the elements (and also across the layers in case of multiple-layers) such that the total discharge of the string equals the specified discharge. Practical application: Simulation of a ditch or horizontal well for which the total discharge is known.

7.4 Line-doublet strings

A line-doublet string may be used to simulate an impermeable or leaky wall. The wall consists of a polyline. The flow through the wall, Q_n is computed as

$$Q_n = H\frac{h_{\text{left}} - h_{\text{right}}}{c_w} \tag{5}$$

where H is the thickness of the aquifer layer, h_{left} and h_{right} are the heads on either side of the wall and c_w [T] is the resistance of the wall defined as

$$c_w = \frac{w_w}{k_w} \tag{6}$$

where w_w is the width of the wall and k_w is the hydraulic conductivity of the wall. The wall becomes impermeable when the resistance is set to infinity.

7.5 Circular area-sinks

Circular area-sinks are used to simulate transient recharge. The recharge is uniform inside the circle but may vary through time.

8 Model commands

There are two types of models that may be created: multi-aquifer models consisting of a sequence of aquifers and leaky layers and quasi-three-dimensional models in which an aquifer is divided into a number of sublayers to simulate three-dimensional flow.

8.1 Regular multi-aquifer model

A multi-aquifer model consists of a regular sequence of aquifer - leaky layer - aquifer - leaky layer - aquifer, etc. For the current implementation, the bottom model layer is an aquifer, which is bounded at the bottom by an impermeable layer. The top model layer may be either an aquifer or leaky layer. The top model boundary may either be impermeable or a fixed water table.

`In []: ml = ModelMaq(kaq=[1],z=[1,0],c=[],Saq=[],Sll=[],topboundary='imp',phreatictop=False, tmin=1,tmax=10,M=20)` where

`kaq` is a list[1] or an array[2] of hydraulic conductivities of the aquifers starting from the top down.

`z` is a list or array of top and bottom elevations of the model layers from the top down. (Note: this may be counter intuitive, so be careful.) The last value is the bottom elevation of the bottom aquifer. The elevations need to be chosen such that the thicknesses of the aquifer layers are all larger than zero; the thickness of a leaky layer may be zero.

`c` is a list or array of the vertical resistances of the leaky layers, starting from the top down. The resistance must always be larger than zero.

`Saq` is a list or array of the specific storage coefficients of the aquifers. When the keyword `phreatictop` is set to `True`, the first value is the phreatic storage coefficient (unless the `topboundary` is set to `'semi'`).

`Sll` is a list or array of the specific storage coefficients of the leaky layers. The first value needs to be the phreatic storage coefficient in case the keyword `phreatictop` is set to `True` and the `topboundary` is set to `'leaky'`.

`topboundary` indicates the boundary condition at the top of the aquifer system. Options are `'standard'` (used to be the somewhat confusing `'impermeable'`, which still works) which means no transient leakage is induced through the top of aquifer layer 0 by the elements in the model, `'leaky'` which means aquifer 0 is covered by a leaky layer and water may flow from the leaky layer into aquifer 0 through the release or increase of storage in the leaky layer, or `'semi'` which means aquifer 0 is covered by a leaky layer which is bounded on top by a fixed head equal to zero.

`phreatictop` is set to `True` when the top model layer contains a phreatic surface. When `topboundary='imp'` this means the the first value of `Saq` is interpreted as the phreatic storage coefficient. When `topboundary='leaky'` this means the the first value of `Sll` is interpreted as the phreatic storage coefficient. When `topboundary='semi'` this keyword is ignored as a phreatic surface is not possible in the top model layer.

`tmin` is the minimum time for which heads can be computed after any change in boundary condition.

[1]A list is a sequence separated by commas and between square brackets, such as: [1,2,3]
[2]Arrays are defined in the `numpy` package

`tmax` is the maximum time for which heads can be computed.

`M` is the number of terms to be used in the numerical inversion algorithm. 20 is usually sufficient. If drawdown curves appear to oscillate, more terms may be needed, but this seldom happens.

As an example, when the number of aquifers is N, then both `kaq` and `Saq` need to contain N values. When `topboundary` is set to `'imp'` then `z` contains $2N$ values, and `c` and `Sll` contain $N-1$ values. When `topboundary` is not `'imp'` then there exists an additional leaky layer on top of aquifer 1, so that `z` contains $2N+1$ values, and `c` and `Sll` contain N values.

8.2 Quasi three-dimensional model

A quasi three-dimensional model consists of one aquifer which is subdivided into an arbitrary number of model layers, each with its own hydraulic conductivity, vertical anisotropy, and storage coefficient. The top and bottom of the aquifer are impermeable.

```
In []: ml = Model3D(kaq=[1,1,1],z=[4,3,2,1],Saq=[0.3,0.001,0.001],kzoverkh=[.1,.1,.1],
     phreatictop=True,tmin=1,tmax=10,M=20)
```
where

`kaq` is a list or an array of horizontal hydraulic conductivities of the aquifer layers starting from the top down. One value may be entered when the hydraulic conductivity is the same in all layers.

`z` is a list or array of top elevations of the model layers from the top down. (Note: this may be counter intuitive, so be careful.) The last value is the bottom elevation of the bottom model layer. The elevations need to be chosen such that the thicknesses of the aquifer layers are all larger than zero.

`Saq` is a list or array of the specific storage coefficients of the model layers. One value may be entered when the specific storage coefficient is the same in all layers. When the keyword `phreatictop` is set to `True`, the first value is the phreatic storage coefficient.

`kzoverkh` is a list or an array of vertical anisotropy values. One value may be entered when the vertical anisotropy is the same in all layers.

`phreatictop` is set to `True` when the top model layer contains a phreatic surface, which means that the first value of `Saq` is interpreted as the phreatic storage coefficient.

`tmin` is the minimum time for which heads can be computed after any change in boundary condition.

`tmax` is the maximum time for which heads can be computed.

`M` is the number of terms to be used in the numerical inversion algorithm. 15 is usually sufficient. If drawdown curves appear to oscillate, 20 terms may need to be used, but this seldom happens. On very rare occasions, more terms may be needed.

As an example, when the aquifer is divided into N model layers, then `z` needs to contain $N+1$ values.

8.3 Cylindrical inhomogeneities

Cylindrical inhomogeneities in all aquifer properties may be added with one of the following commands (depending on whether there is a Multi-Aquifer model or a 3D model)

`In []: CircInhomMaq(model,x0=0,y0=0,R=1,order=1,kaq=[1],z=[1,0],c=[],Saq=[0.001],Sll=[0], topboundary='imp',phreatictop=False,label=None)` or

`In []: CircInhom3D(self,model,x0=0,y0=0,R=1,order=1,kaq=[1,1,1],z=[4,3,2,1], Saq=[0.3,0.001,0.001],kzoverkh=[.1,.1,.1],phreatictop=True,label=None)` where

`model` is the model to which the inhomogeneity is added

`x0,y0` is the location of the center of the cylinder

`R` is the radius of the cylinder

`order` is the order of the inhomogeneity. The boundary condition is met at `2*order+1` control points along the cylindrical inhomogeneity and approximately between the control points.

`label` is a string with a unique label for the cylinder. The default is `None`, which means there is no label. When a label is entered that already exists, TTim throws an error message.

Aquifer parameters are defined as described for the `ModelMaq` and `Model3D` commands.

Cylinders may be nested but may not overlap (TTim does not check whether they overlap, but an invalid solution will be obtained).

9 Analytic element commands

The following elements may be added to the model:

9.1 Wells

There are five types of wells:

```
In []: Well(model,xw=0,yw=0,rw=0.1,tsandQ=[(0.0,1.0)],
     res=0.0,layers=0,rc=None,wbstype='pumping',label=None)
```

```
In []: ZeroMscreenWell(model,xw=0,yw=0,rw=0.1,res=0.0,layers=[0,1],vres=0.0,label=None)
```

```
In []: DischargeWell(model,xw=0,yw=0,rw=0.1,tsandQ=[(0.0,1.0)],res=0.0,layers=0,label=None)
```

```
In []: HeadWell(model,xw=0,yw=0,rw=0.1,tsandh=[(0.0,1.0)],res=0.0,layers=0,label=None)
```

```
In []: ZeroHeadWell(model,xw=0,yw=0,rw=0.1,res=0.0,layers=0,label=None)
```

where

`model` is the model to which the element is added

`xw,yw` is the location of the well

`rw` is the radius of the well

`tsandQ` is a list of `(time,Q)` values, where `Q` is the discharge of the well, positive for taking water out. A 2D array may be entered as well. For example, `tsandQ=[(5,10),(8,0),(14,20)]` means that the discharge is $Q = 10$ starting at time $t = 5$, the discharge is $Q = 0$ starting at time $t = 8$ and $Q = 20$ starting at time $t = 14$.

`tsandh` is a list of `(time,h)` values, similar to entering `tsandQ`. A 2D array may be entered as well.

`res` is the entry resistance of the well, also referred to as the skin effect.

`vres` is the vertical resistance to flow inside the well, which may be zero (for `ZeroMscreenWell` elements only).

`rc` is the radius of the caisson to simulate well bore storage. When the radius is set to `None` or 0, well bore storage is not taken into account.

`layers` is either the layer number where the well is screened or is a list or array with multiple layer numbers in case there are multiple screens. The layer numbers may contain gaps (e.g., `layers=[2,3,6]` may be used for a well screened in layers 2, 3, and 6).

`wbstype` is the well type, either `'pumping'` (default), or `'slug'`.

`label` is a string with the unique label for the well. The default is `None`, which means there is no label. When a label is entered that already exists, TTim throws an error message. It is useful to enter a label to compute, for example, the head inside a well.

9.2 Line-sinks

There are five types of line-sinks:

```
In []: HeadLineSink
     (model,x1=-1,y1=0,x2=1,y2=0,tsandh=[(0.0,1.0)],res=0.0,wh='H',layers=0,label=None)

In []: ZeroHeadLineSink
     (model,x1=-1,y1=0,x2=1,y2=0,res=0.0,wh='H',layers=0,label=None)

In []: ZeroMscreenLineSink
     (model,x1=-1,y1=0,x2=1,y2=0,res=0.0,wh='H',layers=[0,1],vres=0.0,wv=1.0,label=None)

In []: LineSink
     (model,x1=-1,y1=0,x2=1,y2=0,tsandQ=[(0.0,1.0)],res=0.0,wh='H',layers=0,label=None)

In []: MscreenLineSink
     (model,x1=-1,y1=0,x2=1,y2=0,tsandQ=[(0.0,1.0)],res=0.0,wh='H',layers=[0,1],label=None)
```

where

`model` is the model to which the element is added

`x1,y1,x2,y2` are the left and right end points of the line-sink

`tsandQ` is a list of `(time,Q)` values, where `Q` is the discharge of the line-sink, positive for taking water out of the aquifer; the inflow or outflow σ is uniform along the line-sink so that the discharge Q is equal to $Q = \sigma L$. A 2D array may be entered as well. For example, `tsandQ=[(5,10),(8,0),(14,20)]` means that the discharge is $Q = 10$ starting at time $t = 5$, the discharge is $Q = 0$ starting at time $t = 8$ and $Q = 20$ starting at time $t = 14$.

`tsandh` is a list of `(time,h)` values, similar to entering `tsandQ`. A 2D array may be entered as well.

`res` is the entry resistance of the line-sink, also referred to as the stream bed resistance.

`wh` is the distance over which flow enters the line-sink (w in Eq. 4). Values may be `'H'` (default) when the distance is equal to the thickness of the aquifer layer (when flow comes mainly from one side), `'2H'` when the distance is twice the thickness of the aquifer layer (when flow comes from both sides), or a number, for example, the width of the stream that partially penetrates the aquifer.

`layers` is either the layer number where the line-sink is positioned or is a list or array with multiple layer numbers in case the line-sink is open to multiple layers.

`vres` is the vertical resistance inside the line-sink (only for `ZeroMscreenLineSink` elements)

`wv` is the width of a `ZeroMscreenLineSink`, commonly used to simulate a vertical fault.

`label` is a string with the unique label for the line-sink. The default is `None`, which means there is no label. When a label is entered that already exists, TTim throws an error message. It is useful to enter a label to compute, for example, the inflow into a line-sink.

9.3 Strings of line-sinks

The following line-sink string elements have been implemented:

```
In []: HeadLineSinkString
     (model,xy=[(-1,0),(1,0)],tsandh=[(0.0,1.0)],wh='H',res=0.0,layers=0,label=None)
```

```
In []: ZeroHeadLineSinkString(model,xy=[(-1,0),(1,0)],res=0.0,wh='H',layers=0,label=None)
```

```
In []: ZeroMscreenLineSinkString
     (model,xy=[(-1,0),(1,0)],res=0.0,wh='H',layers=[0,‘],vres=0.0,wv=1.0,label=None)
```

```
In []: MscreenLineSinkDitchString
     (model,xy=[(-1,0),(1,0)],tsandQ=[(0.0,1.0)],res=0.0,wh='H',layers=[0,1],label=None)
```

where `xy` is a list or 2D array of `(x,y)` pairs of the vertices of the line-sink string. All other variables are the same as for the line-sink types defined above.

9.4 Strings of line-doublets

The following line-doublet string element has been implemented:

```
In []: LeakyLineDoubletString(model,xy=[(-1,0),(1,0)],res='imp',order=0,layers=0,label=None)
```

where

`model` is the model to which the element is added

`xy` is a list or 2D array of `(x,y)` pairs of the vertices of the string

`res` is the resistance of the leaky wall. Enter `'imp'` for an impermeable wall.

`order` is the order of each leaky wall segment. Condition (5) is applied at `order+1` points along each segment. A good order is 2. It is not advised to increase the order beyond 8 or so. If more control is needed, it is better to use shorter segments.

`layers` is either the layer number where the wall is positioned or is a list or array with multiple layer numbers in case the wall is positioned in multiple layers. The layer numbers may contain gaps (although that doesn't seem to be very useful for leaky walls).

`label` is a string with the unique label for the line-sink. The default is `None`, which means there is no label. When a label is entered that already exists, TTim throws an error message. It is useful to enter a label to compute, for example, the inflow into a line-sink.

9.5 Circular area-sinks

Circular area-sinks are always added to layer 0. They may be added to a model with the following command:

```
In []: CircAreaSink(self,model,xc=0,yc=0,R=0.1,tsandbc=[(0.0,1.0)],label=None)
```

where

`model` is the model to which the element is added

`xc,yc` is the center of the circular area-sink

`R` is the radius of the circular area-sink

`tsandbc` is a list of `(time,N)` values, where `N` is the recharge (dimension: length per time), positive for adding water to the aquifer. A 2D array may be entered as well. For example, `tsandbc=[(5,0.001),(8,0),(14,0.002)]` means that the recharge is $N = 0.001$ (in m/d when units of meters and days are used) starting at time $t = 5$, the recharge is $N = 0$ starting at time $t = 8$ and $N = 0.002$ starting at time $t = 14$.

`label` is a string with the unique label for the area-sink. The default is `None`, which means there is no label. When a label is entered that already exists, TTim throws an error message. It is useful to enter a label to compute, for example, the cumulative recharge of the area-sink.

10 Functions

`In []: ml.solve()` Solve the model.

`In []: ml.head(x,y,t,layers=None)` returns an array with size `(Naq,Ntimes)` or `(Nlayers,Ntimes)` of heads where

- `x,y` are the coordinates of the point where the head is computed.
- `t` is either one value or a list or array with `Ntimes` ordered values. Zero is returned for `t` values outside `tmin` and `tmax`. Note that it is generally much quicker to compute the head for a number of times in one command than to call the `head` function multiple times for the same location but with a different time.
- `layers` is an optional argument. When it is set to `None` (default), the head is computed at all `Naq` aquifers of the model. Optionally, the desired layer number or a list or array of not-necessarily consecutive layer numbers may be specified, in which case the head is computed at `(Nlayers,Ntimes)` points.

`In []: ml.headalongline(self,x,y,t,layers=None)` returns an array with size `(Naq,Ntimes,Npoints)` or `(Nlayers,Ntimes,Npoints)` of heads where

- `x` is an array with the x values of the line.
- `y` is an array of the same length as `x` with the y values of the line, or it is one value in which case all y values will be the same along the line.
- `t` is either one value or a list or array of values and must be ordered. Zero is returned for `t` values outside `tmin` and `tmax`.
- `layers` is an optional argument. When it is set to `None` (default), the head is computed at all `Naq` aquifers of the model. Optionally, the desired layer number or a list or array of not-necessarily consecutive layer numbers may be specified, in which case the head is computed at `(Nlayers,Ntimes,Npoints)` points.

`In []: ml.headgrid(x1,x2,nx,y1,y2,ny,t,layers=None)` returns an array with size `(Naq,Ntimes,ny,nx)` or `(Nlayers,Ntimes,ny,nx)` of heads where

- `x1,x2` are minimum and maximum x-coordinates of the gridding window.
- `nx` is the number of evenly spaced points in x direction where heads are computed.
- `y1,y2` are minimum and maximum y-coordinates of the gridding window.
- `ny` is the number of evenly spaced points in y direction where heads are computed.
- `t` is either one value or a list or array of values and must be ordered. Zero is returned for `t` values outside `tmin` and `tmax`.
- `layers` is an optional argument. When it is set to `None` (default), the head is computed at all `Naq` aquifers of the model. Optionally, the desired layer number or a list or array of not-necessarily consecutive layer numbers may be specified, in which case the head is computed at `(Nlayers,Ntimes,ny,nx)` points.

`In []: e.strength(t)` or `ml.strength(elabel,t)` returns an array with size `(Nlayers,Ntimes)` of the strength of element `e` or with label `elabel` (for a well this is the discharge (units L^3/T), for a line-sink this is the discharge of the line-sink (units L^3/T), for a string of line-sinks this is the discharge of the entire string (units L^3/T)) where `Nlayers` is the number of layers that the element is screened in and where `t` is either one value or a list or array of values and must be ordered. Zero is returned for `t` values outside `tmin` and `tmax`.

`In []: e.headinside(t)` or `ml.headinside(elabel,t)` returns an array with size `(Nlayers,Ntimes)` of the head inside element `e` or with label `elabel` where `Nlayers` is the number of layers that the element is screened in and where `t` is either one value or a list or array of values and must be ordered. Zero is returned for `t` values outside `tmin` and `tmax`.

`In []: ml.writemodel(filename)`. Writes the entire model to the specified filename. This is especially useful if TTim input is created interactively or with a GUI.

11 Interactive plotting

`In []: xsection(ml,x1=0,x2=1,y1=0,y2=0,N=100,t=1,layers=0,color=None,lw=1,newfig=True)` plot heads in a cross-section where the coordinate along the horizontal axis is the distance from `x1,y1` and where

- `x1,x2` are beginning and end x-coordinates of the cross section.
- `y1,y2` are beginning and end y-coordinates of the cross section.
- `N` is the number of points where the head is computed along the cross-section.
- `t` is either one value or a list or array of time values and must be ordered.
- `layers` is either one value or a list or array of layer numbers and must be ordered.
- `color` is the line color. Default is `None` in which case matplotlib will choose a nice color.
- `lw` is the line width. Default is 1.

`newfig` indicates whether a new figure is opened (default `True`) or whether the results are added to the existing open figure when `False`.

`In []: timcontour( ml, xmin, xmax, nx, ymin, ymax, ny, levels = 10, t=0, layers = 0, color = 'k', lw = 0.5, style = 'solid',layout = True, newfig = True, labels = False, labelfmt = '%1.2f')` create contour plot where

`xmin,xmax` are minimum and maximum x-coordinates of the gridding window.

`nx` is the number of points in x direction where heads are computed.

`ymin,ymax` are minimum and maximum y-coordinates of the gridding window.

`ny` is the number of points in y direction where heads are computed.

`levels` is used to specify the head values to contour. There are three options:

- `[hmin,hmax,step]` A list with the minimum, maximum and step size of the head contours you want to see
- `number`. Show `number` contours and let TTim figure out which values to contour.
- `array`. Show contours for the head values in the array.

`t` is the time for which contours are plotted.

`layers` is the layer number for which contours are plotted.

`color` is the color of the contours. When the default `None` is used, the colors will vary with the values.

`lw` the line width of all contours. Default is 0.5.

`style` the line style (`'-'` is a solid line).

`layout` A layout of all elements is shown if this keyword is `True`.

`newfig` A new figure is created when this keyword is `True`. Otherwise, the contours are added to the active figure.

`labels` Labels are shown on the contours if this keyword is `True`.

`labelfmt` Is the format of the numbers of the labels. The default `'%1.2f'` means two numbers behind the decimal.

12 References

M. Bakker. 2013a. Analytic modeling of transient multi-layer flow. In: *Advances in Hydrogeology*, edited by P Mishra and K Kuhlman, Springer, Heidelberg, 95-114.

M. Bakker. 2013b. Semi-analytic modeling of transient multi-layer flow with TTim. *Hydrogeology Journal* 21: 935-943.

M. Bakker and K.L. Kuhlman. 2011. Computational issues and applications of line-elements to model subsurface flow governed by the modified Helmholtz equation. *Advances in Water Resources* 34: 1186-1194.

F.R. De Hoog, J.H. Knight, and A.N. Stokes. 1982. An improved method for numerical inversion of Laplace transforms. *SIAM Journal on Scientic and Statistical Computing*, 3(3):357-366.

C.J. Hemker and C. Maas. 1987. Unsteady ow to wells in layered and ssured aquifer systems. *Journal of Hydrology*, 90:231-249.

G.P. Kruseman and N.A. de Ridder. 1990. Analysis and evaluation of pumping test data. International Institute for Land Reclamation and Improvement (ILRI) Bulletin 11, Wageningen.

K.L. Kuhlman and S.P. Neuman. 2009. Laplace-transform analytic-element method for transient porous-media ow. *Journal of Engineering Mathematics*, 64(2):113-130.

A. Louwyck, A. Vandenbohede, M. Bakker, L. Lebbe. 2012. Simulation of axi-symmetric flow towards wells: A finite-difference approach. *Computers & Geosciences*, 44, 136-145.

S.P. Neuman. 1972. Theory flow in unconfined aquifers considering delayed response of the water table. *Water Resources Research*, 8(4):1031-1045.

13 Acknowledgement

Validation tests 1-3 were carried out by Erik Anderson, who also carefully reviewed this document.