

TTim  
A Multi-Aquifer Transient Analytic Element Model  
Version 0.21

**Mark Bakker**

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

November 6, 2012

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Funding</b>	<b>4</b>
<b>3</b>	<b>License</b>	<b>4</b>
<b>4</b>	<b>TTim website</b>	<b>4</b>
<b>5</b>	<b>Installation</b>	<b>4</b>
5.1	Installation instructions for Windows . . . . .	5
5.2	Installation on Mac OS X . . . . .	5
5.3	Running TTim on a platform other than Windows or Mac . . . . .	5
5.4	Release history . . . . .	5
<b>6</b>	<b>Main approximations</b>	<b>5</b>
<b>7</b>	<b>Types of elements and their practical application</b>	<b>6</b>
7.1	Wells . . . . .	6
7.2	Line-sinks . . . . .	7
7.3	Line-sink strings . . . . .	8
7.4	Line-doublet strings . . . . .	8
<b>8</b>	<b>Model commands</b>	<b>8</b>
8.1	Regular multi-aquifer model . . . . .	9
8.2	Quasi three-dimensional model . . . . .	10
<b>9</b>	<b>Analytic element commands</b>	<b>10</b>
9.1	Wells . . . . .	10
9.2	Line-sinks . . . . .	11
9.3	Strings of line-sinks . . . . .	12
9.4	String of line-doublets . . . . .	13
<b>10</b>	<b>Functions</b>	<b>13</b>
<b>11</b>	<b>Interactive plotting</b>	<b>15</b>
<b>12</b>	<b>Validation tests</b>	<b>16</b>
12.1	Test 1: Comparison with the MLU model . . . . .	16
12.2	Running the model . . . . .	16
12.3	Test 2: Comparison of line-sink element with multiple wells . . . . .	19
12.4	Test 3: Comparison with a MODFLOW model . . . . .	20
12.5	Test 4: Comparison with analytic solution for unconfined flow . . . . .	22
<b>13</b>	<b>References</b>	<b>25</b>

**14 Acknowledgement**

**25**

## 1 Introduction

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

## 2 Funding

Development of TTim 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. Papadopoulos and Associates in Bethesda, MD. Version 0.01 was developed in collaboration with S.S. Papadopoulos and Associates in Bethesda, MD, and Layne Hydro in Bloomington, IN.

## 3 License

Copyright ©2010-2012, Mark Bakker

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 4 TTim website

TTim is available from [www.ttlim.googlecode.com](http://www.ttlim.googlecode.com).

## 5 Installation

TTim requires installation of the free and open source software Python, and the packages numpy, scipy, and matplotlib. TTim uses FORTRAN extensions which means that TTim is both platform dependent and Python version dependent. The installer name indicates the Python version that is required. For example, the file `TTim-0.02.py27.win32.exe` is TTim version 0.02 for Python 2.7 on Windows.

## 5.1 Installation instructions for Windows

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. Two of the best options on Windows are PythonXY and the free Enthought Python Distribution (EPDFree). The installation is straightforward. When you want to use PythonXY, download the PythonXY installer from the 'Downloads' tab on <http://pythonxy.googlecode.com>. Once downloaded, install PythonXY. When you want to use the Enthought version, download the EPDFree installer from [http://enthought.com/products/epd\\_free.php](http://enthought.com/products/epd_free.php). Once downloaded, install EPDFree. Next, download and install TTim. A nice and free editor for Python scripts is Komodo Edit, available from [http://www.activestate.com/komodo\\_edit/downloads/](http://www.activestate.com/komodo_edit/downloads/).

## 5.2 Installation on Mac OS X

For Mac OS X, install the EPDFree distribution of Enthought or, for a small fee (free for Academia), the full Enthought distribution. Download the Enghought distribution from <http://www.enthought.com>. Next, download and install TTim. A nice and free editor for Python scripts is Komodo Edit, available from [http://www.activestate.com/komodo\\_edit/downloads/](http://www.activestate.com/komodo_edit/downloads/)

## 5.3 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`.

## 5.4 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 0.3. November 2012.

# 6 Main approximations

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, where the first leaky layer is on top of the first aquifer layer.

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

Two kinds of elements are implemented: wells and line-sinks. Each kind of element has 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: **MscreenWell**. 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:

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

where  $r_w$  is the radius of the well,  $H$  is the aquifer thickness,  $c$  is the entry resistance (with units of time), and  $h_{\text{out}}$  and  $h_{\text{in}}$  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

$$s_s = \frac{Q}{2\pi T} c_s \quad (2)$$

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

$$c_s = \frac{k}{r_w} c \quad (3)$$

## 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: **MscreenLineSink**. 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

$$Q = Lw \frac{h_{\text{out}} - h_{\text{in}}}{c} \quad (4)$$

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  $h_{\text{out}}$  and  $h_{\text{in}}$  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

$$Q_n = H \frac{h_{\text{left}} - h_{\text{right}}}{c_w} \quad (5)$$

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

$$c_w = \frac{w_w}{k_w} \quad (6)$$

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.

## 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.

```
>>> m1 = ModelMaq(kaq=[1],z=[1,0],c=[],Saq=[],S11=[],topboundary='imp',phreatictop=False,
    tmin=1,tmax=10,M=20) where
```

**kaq** is a list<sup>1</sup> or an array<sup>2</sup> 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'**).

**S11** 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 1 by the elements in the model, **'leaky'** which means aquifer 1 is covered by a leaky layer and water may flow from the leaky layer into aquifer 1 through the release or increase of storage in the leaky layer, or **'semi'** which means aquifer 1 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 **S11** 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.

**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.

---

<sup>1</sup>A list is a sequence separated by commas and between square brackets, such as: [1,2,3]

<sup>2</sup>Arrays are defined in the **numpy** package

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 **S11** 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 **S11** 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.

```
>>> m1 = 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.

## 9 Analytic element commands

The following elements may be added to the model:

### 9.1 Wells

There are five types of wells:

```
>>> Well(model,xw=0,yw=0,rw=0.1,tsandQ=[(0.0,1.0)],
    res=0.0,layers=1,rc=None,wbstype='pumping',label=None)
```

```
>>> ZeroMscreenWell(model,xw=0,yw=0,rw=0.1,res=0.0,layers=[1,2],vres=0.0,label=None)
>>> DischargeWell(model,xw=0,yw=0,rw=0.1,tsandQ=[(0.0,1.0)],res=0.0,layers=1,label=None)
>>> HeadWell(model,xw=0,yw=0,rw=0.1,tsandh=[(0.0,1.0)],res=0.0,layers=1,label=None)
>>> ZeroHeadWell(model,xw=0,yw=0,rw=0.1,res=0.0,layers=1,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).

`label` is a alpha 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:

```
>>> HeadLineSink
      (model,x1=-1,y1=0,x2=1,y2=0,tsandh=[(0.0,1.0)],res=0.0,wh='H',layers=1,label=None)
>>> ZeroHeadLineSink
      (model,x1=-1,y1=0,x2=1,y2=0,res=0.0,wh='H',layers=1,label=None)
>>> ZeroMscreenLineSink
      (model,x1=-1,y1=0,x2=1,y2=0,res=0.0,wh='H',layers=[1,2],vres=0.0,wv=1.0,label=None)
```

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

>>> MscreenLineSink
      (model,x1=-1,y1=0,x2=1,y2=0,tsandQ=[(0.0,1.0)],res=0.0,wh='H',layers=[1,2],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  $\sigma$  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. 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).

`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:

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

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

>>> ZeroMscreenLineSinkString
      (model,xy=[(-1,0),(1,0)],res=0.0,wh='H',layers=[1,2],vres=0.0,wv=1.0,label=None)
```

```
>>> MscreenLineSinkDitchString
```

```
(model,xy=[(-1,0),(1,0)],tsandQ=[(0.0,1.0)],res=0.0,wh='H',layers=[1,2],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 String of line-doublets

The following line-doublet string element has been implemented: The following line-sink string elements have been implemented:

```
>>> LeakyLineDoubletString(model,xy=[(-1,0),(1,0)],res='imp',order=0,layers=1,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.

## 10 Functions

```
>>> m1.solve() Solve the model.
```

```
>>> m1.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.

>>> `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.

>>> `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.

>>> `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`.

>>> `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`.

>>> `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

`>>> xsection(ml,x1=0,x2=1,y1=0,y2=0,N=100,t=1,layers=1,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`.

`>>> timcontour( ml, xmin, xmax, nx, ymin, ymax, ny, levels = 10, t=0, layers = 1, 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.

Table 1: Aquifer data for Validation Test 1

	kaq (m/day)	bottom elev. (m)	top elev. (m)	c (days)	Saq (1/m)	Sll (1/m)
Aquifer 1	1	2	3	10	0.3	0.001
Leaky layer 1		1	2			
Aquifer 2	5	0	1		0.01	

## 12 Validation tests

Four code validation tests are described here, along with results. The first test is a check of the inversion algorithm used by TTim, by comparison with results from a commercially available software package, MLU ([www.microfem.com/products/mlu.html](http://www.microfem.com/products/mlu.html)). The second test is an internal check of the TTim line-sink elements: TTim results for a discharge-specified line-sink are compared to results where the line-sink is approximated as a string of wells. The third test compares TTim results with MODFLOW results for an injection well operating near a crack or fracture connecting two aquifers. The final test is a comparison between TTim results and the analytical solution of Neuman (1972) for drawdown near a well in an unconfined aquifer.

Model files for all of the tests are provided with this documentation, each file named according to the validation test number (ttimtest1.py, ttimtest2.py, ttimtest3.py, ttimtest4.py). The model data sets may be viewed and modified using any ASCII editor. Model data may be entered interactively at the IPython prompt, or the data files may be run from the prompt, as will be explained for test 1, below.

### 12.1 Test 1: Comparison with the MLU model

TTim results for radial flow to a pumping well screened in one or more aquifers in a multi-aquifer system are exact in the Laplace domain. TTim uses the algorithm of DeHoog et al. (1982) to invert numerically the solution to the time domain. This validation test compares results from TTim to results from the commercial software MLU. MLU uses a similar formulation for radial flow as TTim, but the numerical inversion is performed with the Stehfest algorithm. This validation test verifies the general implementation of pumping wells in TTim and the performance of the inversion algorithm used in TTim vs. the inversion algorithm used in MLU. A free version of MLU (MLU Lite) is available for modeling two-aquifer systems; MLU Lite was used for the comparison described here.

The aquifer system examined consists of two aquifer layers separated by a leaky layer. Aquifer properties specified in this validation test are provided in Table 1. Comparisons of results for a well pumping at  $1 \text{ m}^3/\text{day}$  in aquifer 1 are presented in Figure 1, which includes plots of drawdown at increasing radial distances from the well. The well is given a negligible radius so that MLU and TTim solve the same mathematical problem.

### 12.2 Running the model

It is explained here how to run the TTim model for validation test 1, both by entering data at the IPython prompt and by reading the model file included with this documentation.

In this manual, TTim is run from the IPython prompt. On Windows, when using the PythonXY



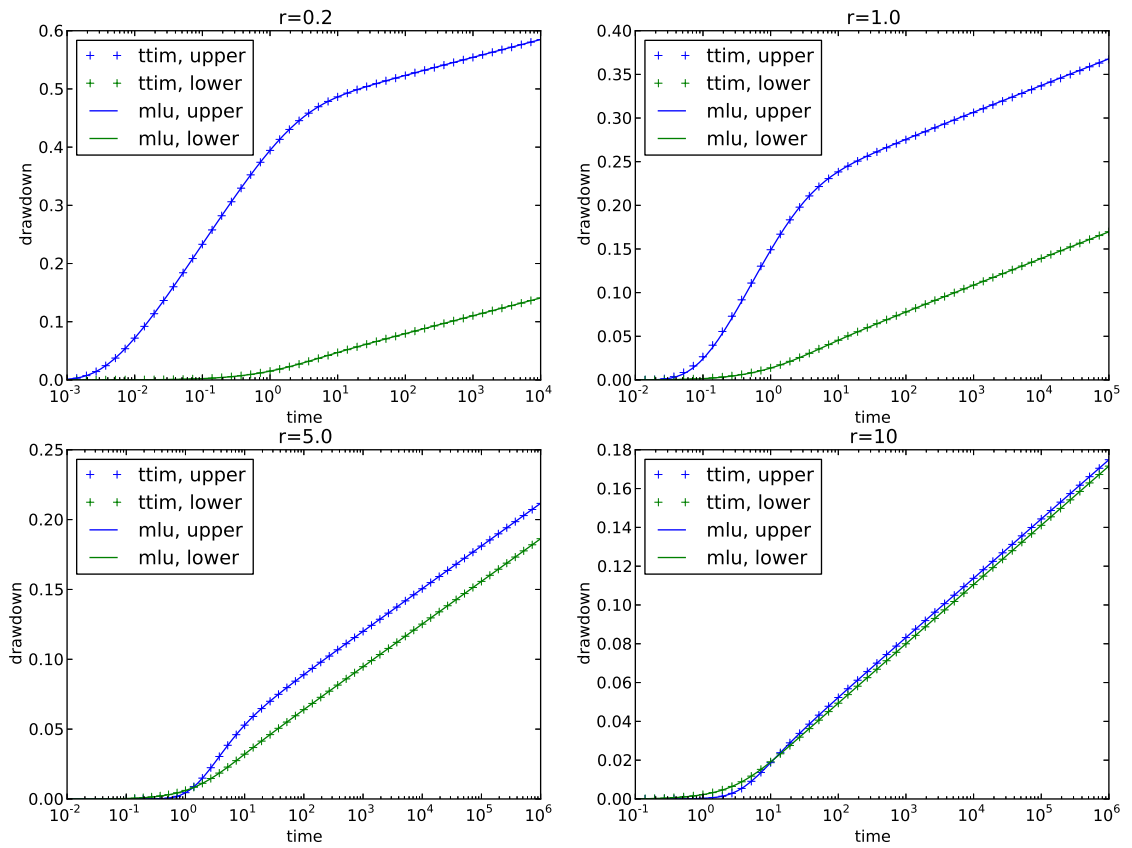


Figure 1: Comparison of drawdown between TTim and MLU at four radial distances: TTim results for the upper aquifer (blue +) and lower aquifer (green +), and MLU results for the upper aquifer (blue line) and lower aquifer (green line). The horizontal axis is in days and the vertical axis in meters.

distribution, go to Start and choose the PythonXY option with a little triangle behind it. When the mouse is positioned on the triangle, multiple options appear. Choose Enhanced Console and then IPython. This starts the IPython interface, which is used in this manual. At startup, you will see something like this:

```
Executing Python(x,y) 2.6.5.3 profile startup script: default.py
Loading NumPy
Loading SciPy
Importing all NumPy functions, modules and classes
Logging to C:\Documents and Settings\erik\.xy\logs\2010-11-10.py
```

```
Python 2.6.5 (r265:79096, Mar 19 2010, 21:48:26) [MSC v.1500 32
bit (Intel)] Type "copyright", "credits" or "license" for more
information.
```

```
IPython 0.10 -- An enhanced Interactive Python. ?           ->
Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system. object?   -> Details about
'object'. ?object also works, ?? prints more.
```

```
IPython profile: xy
```

```
Welcome to pylab, a matplotlib-based Python environment.
For more information, type 'help(pylab)'.
```

```
In [1]:
```

At the Python command prompt, import the ttim.py file:

```
In [1]: from ttim import *
```

Now we can begin to enter model data. First define the model:

```
In [2]: m1 = ModelMaq(kaq=[1.0,5.0],z=[3,2,1,0],c=[10.],
Saq=[0.3,0.01], S11=[0.001], tmin=0.001, tmax=1000000.0, M=20)
```

Next enter the data for the pumping well:

```
In [3]: Well(m1,xw=0.,yw=0,rw=1e-5,tsandQ=[(0,1)],layers=[1],label='well 1')
```

Finally solve the model:

```
In [4]: m1.solve()
```

TTim responds with information indicating that a solution is obtained:

```
self.Neq 0
No unknowns. Solution complete
```

Table 2: Aquifer data for Validation Test 2

	kaq (m/day)	bottom elev. (m)	top elev. (m)	c (days)	Saq (1/m)	Sll (1/m)
Aquifer 1	1	2	3	1000	0.3	0.001
Leaky layer 1		1	2			
Aquifer 2	5	0	1		0.003	

Once the solution is obtained you can use the TTim functions described in this manual to evaluate heads, and the interactive plotting functions to evaluate model results graphically.

Alternatively, model data may be stored in a file with extension `.py`, which can be run from the IPython prompt. For validation test 1, the model data is contained in the file `ttimtest1.py` and is stored, for example, in the directory `c:\ttim_model`. First, change the working directory:

```
In [1]: cd c:\ttim_model
```

Then run the `ttimtest1` file:

```
In [2]: run ttimtest1
```

The blue crosses in Fig. 1 may be obtained, for example, as follows. Create an array of 50 evenly spaced in log10 space from  $1E-3$  till  $1E4$

```
In [3]: t = logspace(-3,4,50)
```

Compute the heads at  $(x,y) = (1,0)$  for all aquifers and all times in `t`

```
In [4]: h = m1.head(1,0,t)
```

The array `h` will have 2 rows (one for each layer) and 50 columns (one for each time). Plot the drawdown (minus the head) vs. time for aquifer 1 (rows and columns in arrays start to count at 0) with blue crosses and a log time axis

```
In [5]: semilogx(t,-h[0], 'b+')
```

### 12.3 Test 2: Comparison of line-sink element with multiple wells

This test is a validation of the line-sink function in TTim. A comparison is made between TTim results for a line-sink in the top aquifer of a two-aquifer system, and TTim results where the line-sink is modeled as a series of wells of specified discharge, screened in the upper aquifer. This is a discrete form of a line-sink; increasing the number of wells, while maintaining the total discharge of the wells results in a better approximation of the line-sink. In the limit as the distribution of wells becomes continuous, the exact solution for the line-sink is produced.

Properties of the aquifer used for the example are shown in Table 2. The strength of the line-sink is  $1 \text{ m}^2/\text{day}$ , and the line-sink is 10 m long for a total discharge of  $10 \text{ m}^3/\text{day}$ . The model of the line-sink is approximated with a model of 10 wells spaced 1 m apart, each pumping at a rate of  $1 \text{ m}^3/\text{day}$ .

Table 3: Aquifer data for TTim model, Validation Test 3

	kaq (m/day)	bottom elev. (m)	top elev. (m)	c (days)	Saq (1/m)	Sll (1/m)
Aquifer 1	1	2	3	10	0.03	0.001
Leaky layer 1		1	2			
Aquifer 2	5	0	1		0.03	

Table 4: Aquifer data for MODFLOW model, Validation Test 3

	$k_x$ (m/day)	$k_y$ (m/day)	$k_z$ (m/day)	Saq (1/m)
Layer 1	1	1	1	0.03
Layer 2	0.1	0.1	0.1	0.001
Layer 3	5	5	5	0.03

Results from the two models are compared at various locations near and far from the line-sink in Fig. 2. Only the upper-left hand graph in Fig. 2 indicates a difference between results of the two models; we note that results presented for the multiple well model are for a point mid-way between two wells, where the difference should be greatest. Contours of drawdown in the upper aquifer layer at  $t = 40$  days are shown in the bottom-right hand corner of Fig. 2.

## 12.4 Test 3: Comparison with a MODFLOW model

Validation test 3 investigates an injection well pumping at a rate of  $10 \text{ m}^3/\text{day}$  into a two aquifer and one leaky layer system. The well lies near a vertical, linear crack through the aquifer system that hydraulically links the two aquifer layers. The well is at the origin of the system with the crack lying 5 m to the west of the well and extending 10 m in a north-south direction. The tips of the crack are at coordinates  $(-5 \text{ m}, -5 \text{ m})$  and  $(-5 \text{ m}, 5 \text{ m})$ .

Properties of the aquifer defined in the TTim model are presented in Table 3. The crack is modeled with 10 MscreenLineSinks, each of 1 m in length, and each with a total strength of zero. There is no vertical resistance to flow along the line-sink, and thus the drawdown along the line-sink is the same in the upper and lower layers at the control points. Note that the crack could have also been modeled in an alternate fashion as 10 HconnLineSinks with zero specified resistance.

The MODFLOW model consists of a three-layer system; properties of the aquifer system defined in the MODFLOW model are presented in Table 4. The layers are each 1 m thick, as in the TTim model. The model consists of a circular domain 200 m in diameter, bounded by no-flow cells. The majority of the domain consists of cells 0.2 m by 0.2 m, with the maximum cell size being 2.0 m by 2.0 m.

The crack is modeled with MODFLOW as a thin zone 0.4 m (2 cells) wide, of different conductivity and storativity cutting through all three model layers. Properties of the crack are presented in Table 5; the assigned properties are intended to simulate the conditions of the TTim model. For example, the high vertical conductivity in each of the layers of the crack is defined to simulate negligible vertical resistance

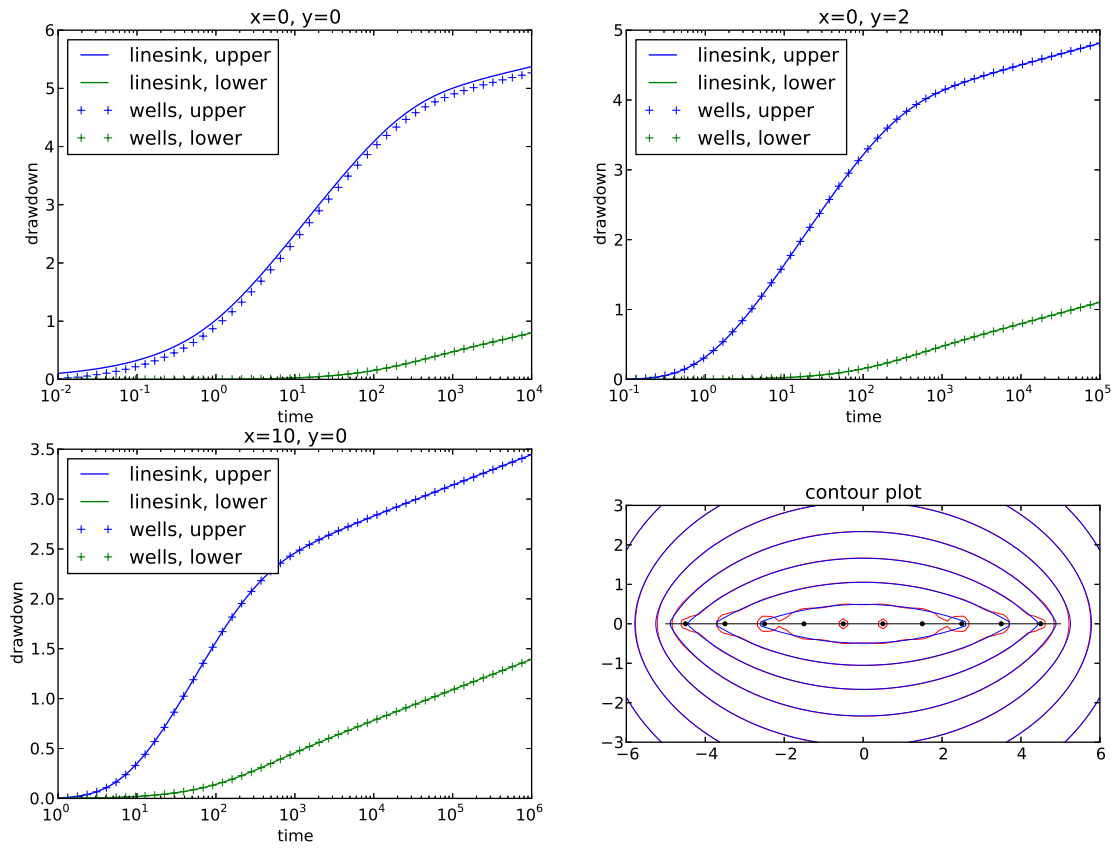


Figure 2: Comparison of drawdown for the line-sink and the discretized line-sink at three (x,y) locations at a contour plot. Results for the upper aquifer are in blue and results for the lower aquifer are in green. The horizontal axis is in days and the vertical axis in meters, except for the contour plot in the bottom right-hand corner where both axes are meters.

Table 5: Crack data for MODFLOW model, Validation Test 3

	$k_x$ (m/day)	$k_y$ (m/day)	$k_z$ (m/day)	Saq (1/m)
Layer 1	1	1	100	1e-05
Layer 2	0.1	0.1	100	1e-05
Layer 3	5	5	500	1e-05

Table 6: Aquifer data for comparison to Neuman solution, Validation Test 4

	kaq (m/day)	thickness (m)	Saq ( $\sigma = 0.1$ )	Saq ( $\sigma = 0.01$ )	Saq ( $\sigma = 0.001$ )
Layer 1	1	0.01	0.1	0.1	0.1
Layer 2-11	1	1	0.001 m <sup>-1</sup>	0.0001 m <sup>-1</sup>	0.00001 m <sup>-1</sup>

within the crack.

Comparisons of drawdown at the north tip, the center, and the south tip of the crack are shown in Fig. 3. The graphs show little head difference between the upper and lower aquifers in both the TTim and MODFLOW models, indicating that the crack is modeled accurately in both models. In the Figures, the MODFLOW results begin to diverge from the TTim results after approximately 40 days: the slope of the drawdown curve increases. This is due to the effects of the finite domain in the MODFLOW model; the no-flow condition specified at the perimeter of the model begin to affect the results near the crack.

Drawdown, at  $t = 10$  days, along an west to east section passing through both the crack and the well is shown in Fig. 3. The crack is located at 5 m and the well at 10 m from the left end of the section.

## 12.5 Test 4: Comparison with analytic solution for unconfined flow

Consider flow to a pumping well in an unconfined aquifer. The drawdown vs. time curves will show the effect of the delayed response of the water table. Neuman (1972) published a transient three-dimensional well solution for the case that the aquifer thickness may be approximated as constant, which is reasonable when the drawdown is small compared to the saturated thickness. Phreatic storage is taken into account through the boundary condition at the top of the aquifer. The same problem may be solved with TTim. An unconfined aquifer is divided into ten model layers of 1 meter thickness and with elastic storage. A thin eleventh layer is added on top of the aquifer and has phreatic storage. The values are summarized in Table 6. One well is screened in layers 2 through 11. As the `Well` element is used, the discharge is specified for each layer to be 1 m<sup>3</sup>/d for a total of 10 m<sup>3</sup>/d. This facilitates comparison with the solution of Neuman, who specifies a uniform inflow along the well face. (In reality it would be better to use an `MscreenWell` so that the head is uniform along the well screen and the inflow varies.) The effect of the delayed response of the water table is clearly visible in Fig. 4, but it is noted that this effect is much less pronounced when the curves are plotted on a linear scale rather than a log scale.

A comparison is made for several curves of Fig. 2 in Neuman (1972). This graph (shown in Fig. 4) shows dimensionless drawdown vs. dimensionless time, both on a log scale, at the bottom of the aquifer at a

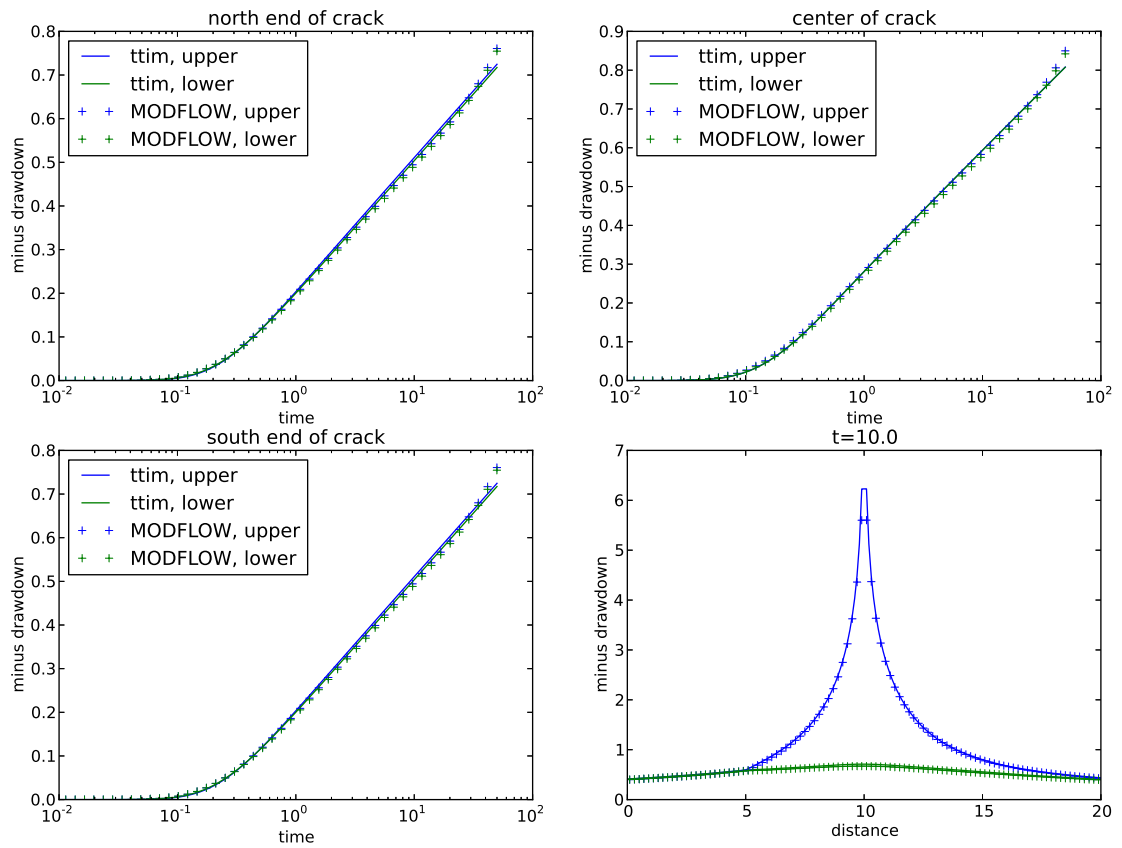


Figure 3: Drawdown at the center of the crack at three different locations and heads along a cross-section. Results for the upper aquifer are in blue and for the lower aquifer are in green. The horizontal axis is in days and the vertical axis in meters, except for the cross-sectional figure, where both axes are in meters.

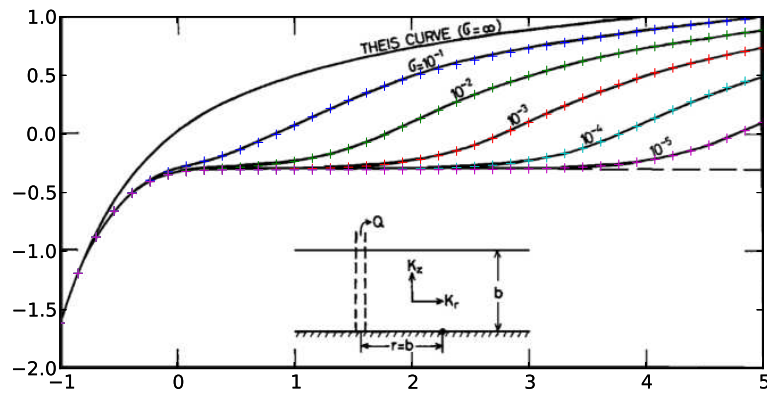


Figure 4: Dimensionless drawdown vs. dimensionless time at one aquifer thickness from a well in an unconfined aquifer.  $\sigma = S_{\text{elastic}}/S_{\text{phreatic}}$ . Black lines are copied from Fig. 2 of Neuman (1972), while the crosses are computed with TTim.



distance of one aquifer thickness from the well for an isotropic aquifer. The curves represent different values of  $\sigma$ , defined as the ratio of the total elastic storage of the aquifer divided by the phreatic storage. The same problem is solved with TTim for three values of  $\sigma$ . The crosses in Fig. 4 represent the drawdown computed with TTim in the bottom layer of the model. This test validates the accuracy of TTim for modeling quasi three-dimensional flow using `Model3D` in combination with phreatic storage and a `Well` element screened in multiple aquifer layers.

## 13 References

- 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 Scientific and Statistical Computing*, 3(3):357-366.
- C.J. Hemker and C. Maas. 1987. Unsteady flow to wells in layered and fissured 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 flow. *Journal of Engineering Mathematics*, 64(2):113-130.
- S.P. Neuman. 1972. Theory flow in unconfined aquifers considering delayed response of the water table. *Water Resources Research*, 8(4):1031-1045.

## 14 Acknowledgement

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