Theis
a Python class implementing the Theis solution for the pressure perturbation due to injection into an aquifer
Version 0.2

Karl W. Bandilla

May 7, 2010

© Karl W. Bandilla 2010

KWBandilla@gmail.com

1 Introduction

Theis is a class programmed in the scripting language Python. The purpose of the class is to compute the pressure disturbance caused by injecting a fluid into an aquifer bounded by two impervious layers. The analytic solution implemented in this class is based on Theis (1935) [1]. The units of the input variables are assumed to be consistent (e.g., all times in seconds, and all lengths in meters). The validity of input variables is not checked within this class, so bad input values (e.g., strings instead of numbers, negative values, ...) are going to lead to crashes.

2 License

Theis is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software

Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

3 Release history

Version 0.1 January 2010: first release

Version 0.2 May 2010: output in either head or pressure; customize warning messages; allow time, distance and pressure/head to be entered as single values or lists.

4 Installation

Theis_0_2.py and *ExampleTheis_0_2.py* can be freely downloaded from code.google.com/p/camelotpy. In order to use *Theis_0_2.py* Python 2.5 needs to be installed on the computer. Later versions of Python may be used, if they are supported by the *SciPy* module. Python and all needed third-party modules are freely available online. It is suggested that Python is installed from a package such as *PythonXY* which includes all third-party modules needed for *Theis_0_2.py*. Alternatively, Python can be installed from www.python.org. In this case the package *SciPy* needs to be installed separately. *Theis_0_2.py* is part of the *CAMELOT* framework, and is thus copied to the "site-packages" folder when *CAMELOT* is installed.

5 Class structure

5.1 Global variables

The following variables store the main domain properties, and are available throughout the class:

- T: hydraulic transmissivity [L^2/T] (float).
- S: storativity (volume of water that a unit decline of head releases from storage in a vertical prism of the aquifer of unit cross section) [-] (float).
- Q: volumetric injection rate [L^3/T] (float). Positive into the aquifer.
- *InjectionEnd*: end of injection [T] (float).
- *HorP*: return value as piezometric head or as pressure (string). 'H' for head; 'P' for pressure.
- *FreshWDensity*: density of fresh water [M/L^3] (float).
- *Gravity*: gravitational acceleration [L/T^2] (float).
- *[WarnNoRadius]*: value that is returned if function *GetRadiusForHeadTime* cannot find a radius. The first value in the list determines what type of value is returned: 'i' for integer flag; 's' for string. The second value in the list is the value that is returned.
- *[WarnGTsteadystate]*: value that is returned if function *GetTimeForRadiusHead* cannot find a time. The first value in the list determines what type of value is returned: 'i' for integer flag; 's' for string. The second value in the list is the value that is returned.

The following variables are dimensionless variables derived from the variables above. These variables are available throughout the class, but are intended for internal use only.

- $uFactor = \frac{S}{4T}$. Factor to convert to dimensionless time/space.
- $HeadFactor = \frac{Q}{4\pi T}$. Factor to convert from dimensionless head to regular head. For pressure: $HeadFactor = \frac{Q}{4\pi T} \cdot Gravity \cdot FreshWDensity$

5.2 Functions

The *Theis* class consists of several functions. The class also references two generic modules (*SciPy.special* and *math*). The functions are:

- *SetRequiredProperties(Q, S, T, InjectionEnd = -1, HeadOrPressure = 'H', FreshWDensity = 1000.0, Gravity = 9.81, WarnNoRadius=['s', 'no contour'], WarnGTsteadystate=['s', 'pressure not reached'])*: takes the volumetric injection rate (*Q*), the storativity of the aquifer (*S*), the hydraulic transmissivity of the aquifer (*T*), the time at which injection ends (*InjectionEnd*, optional parameter, default set to -1), a flag to specify head or pressure output (*HeadOrPressure*, optional parameter, default set to 'H'), density of fresh water (*FreshWDensity*, optional parameter, default set to 1000.0), gravitational acceleration (*Gravity*, optional parameter, default set to 9.81), output value if no radius is found (*WarnNoRadius*, optional parameter, default set to ['s', 'no contour']), and output value if pressure or head is not reached (*WarnGTsteadystate*, optional parameter, default set to ['s', 'pressure not reached']). This function stores the input parameters as global variables, and computes and stores the dimensionless variables (*uFactor* and *HeadFactor*). This function does not return any value upon completion.

- *SetProperty(proptype, value)*: sets individual model properties specified by *proptype* to the value given in *value*. The following properties can be changed using *SetProperty*: '*T*' aquifer transmissivity (float), '*S*' aquifer storativity (float), '*Q*' volumetric injection rate (float), '*InjectionEnd*' end time of injection (float), '*HeadOrPressure*' head or pressure output (string), '*FreshWDensity*' density of fresh water (float), '*Gravity*' gravitational acceleration (float), '*WarnNoRadius*' output value if no radius is found (list of one string and either a string or a float), and '*WarnGTsteadystate*' output value if pressure or head is not reached (list of one string and either a string or a float). This function recomputes the dimensionless variables *uFactor* and *HeadFactor*. *SetProperty* does not return any value upon completion.

- *GetHeadForRadiusTime(radius, time)*: gives the head or pressure for a given radius and time. *radius* and *time* can be given either as single floats or they can be given as lists of floats. If both *radius* and *time* are given as single floats, the function returns the value of head or pressure upon completion. If one or both are given as a list of floats, the function returns a list with one entry for each combination of values in the *radius* and *time* lists. The list entries consist of the value of time,

radius and head or pressure.

- *GetRadiusForHeadTime(head, time)*: gives the radius for a given value of head or pressure and time. This function takes longer than *GetHeadForRadiusTime* as finding the radius is an iterative process requiring multiple head computations. *head* and *time* can be given either as single floats or they can be given as lists of floats. If both *head* and *time* are given as single floats, the function returns the value of radius upon completion. If one or both are given as a list of floats, the function returns a list with one entry for each combination of values in the *head* and *time* lists. The list entries consist of the value of time, radius and head or pressure. If the given value for head or pressure is no reached, the value specified in *WarnNoRadius* is returned instead of the radius.

- *GetTimeForRadiusHead(radius, head)*: gives the time for a given value of head or pressure to be reached at a given radius. This function takes longer than *GetHeadForRadiusTime* as finding the time is an iterative process requiring multiple head computations. *radius* and *head* can be given either as single floats or they can be given as lists of floats. If both *radius* and *head* are given as single floats, the function returns the value of time upon completion. If one or both are given as a list of floats, the function returns a list with one entry for each combination of values in the *radius* and *head* lists. The list entries consist of the value of time, radius and head or pressure. If the given value for head or pressure is not reached, the value specified in *WarnGTsteadystate* is returned instead of the time.

- *ListOrSingle(inValues)*: determines if the 'inValues' is a list or a single value. The function returns two values: the first value is a list (if 'inValues' is a list the original list is returned; otherwise a list with a single entry with the value 'inValues' is returned), the second is a boolean with the value 'True' if 'inValues' is a list and 'False' in 'inValues' is a single value.

- *Interpolate(x1, y1, x2, y2, yknown)*: linear interpolation of the x value between point 1 at (*x1*, *y1*) and point 2 at (*x2*, *y2*) at y location *yknown*. Returns the interpolated x value.

6 Example

This example shows how the *Theis* class is used to compute the heads, contours and times to reach given heads for continuous injection and injection shut-off. This script is given in *ExampleTheis_0_2.py*, and the module *Theis_0_2.py* should either be in same directory as *ExampleTheis_0_2.py* or be located within the Python search path.

```
import Theis_0_2 as th

# model properties
Q = 0.1 #m^3/s, injection rate
S = 2e-3 #storativity of aquifer
T = 5e-3 #m2/s, aquifer transmissivity
#Specify when to generate results (time from injection start [s])
times = [1e2, 1e4, 1e6, 1e9]
#Specify what pressure head [m] to generate radii for
contours = [0.1, 10.0, 25.0]
#Specify distance from well [m]
radius = [10.0, 100.0, 5000.0, 10000.0, 100000.0]

#initialize the model
model = th.Theis()
model.SetRequiredProperties(Q, S, T)

#Generate heads for specified locations and times
print 'Head for given time and distance'
print 'time [s], distance [m], head [m]'
for tim in times:
    for dis in radius:
        print tim, dis, model.GetHeadForRadiusTime(dis, tim)

#Generate head contours for specified pressures and times
print 'Contour radius for given head and time'
print 'time [s], distance [m], head [m]'
for tim in times:
    for con in contours:
        print tim, model.GetRadiusForHeadTime(con, tim), con
```

```
#Generate times to reach head at specified distance
model.SetProperty('InjectionEnd', 1e9)
print 'Time to reach given head at given radius'
print 'time [s], distance [m], head [m]'
for rad in radius:
    for con in contours:
        print model.GetTimeForRadiusHead(rad, con), rad, con

#Generate heads for specified locations and times with an injection
#shut-off after 1e7 seconds
model.SetProperty('InjectionEnd', 1e7)
model.SetProperty('HeadOrPressure', 'P')
model.SetProperty('FreshWDensity', 1000.0)
print 'Head for given time and distance with shut-off after 1e7 s'
print 'using list input and output'
print 'time [s], distance [m], pressure [Pa]'
print model.GetHeadForRadiusTime(radius, times)
```

Running the *ExampleTheis_0_2.py* script should lead to the following output:

```
Head for given time and distance
time [s], distance [m], head [m]
100.0 10.0 2.90127358863
100.0 100.0 6.61602153439e-006
100.0 5000.0 0.0
100.0 10000.0 0.0
100.0 100000.0 0.0
10000.0 10.0 10.0769578718
10000.0 100.0 2.90127358863
10000.0 5000.0 1.69251596437e-111
10000.0 10000.0 0.0
10000.0 100000.0 0.0
1000000.0 10.0 17.4047386244
1000000.0 100.0 10.0769578718
1000000.0 5000.0 0.0396533233578
1000000.0 10000.0 6.61602153439e-006
1000000.0 100000.0 0.0
```

```
1000000000.0 10.0 28.3987567081
1000000000.0 100.0 21.0694022949
1000000000.0 5000.0 8.62102111549
1000000000.0 10000.0 6.42656451963
1000000000.0 100000.0 0.34916037594
Contour radius for given head and time
time [s], distance [m], head [m]
100.0 42.6710443425 0.1
100.0 1.02449740378 10.0
100.0 0.0091985322188 25.0
10000.0 426.710443425 0.1
10000.0 10.2449740378 10.0
10000.0 0.091985322188 25.0
1000000.0 4267.10443425 0.1
1000000.0 102.449740378 10.0
1000000.0 0.91985322188 25.0
1000000000.0 134937.69026 0.1
1000000000.0 3239.74525288 10.0
1000000000.0 29.0883129419 25.0
Time to reach given head at given radius
time [s], distance [m], head [m]
5.4920913884 10.0 0.1
9527.43902439 10.0 10.0
118185148.854 10.0 25.0
549.203106293 100.0 0.1
952743.902439 100.0 10.0
11818514885.4 100.0 25.0
1373007.76573 5000.0 0.1
2381871102.66 5000.0 10.0
2.95462872135e+013 5000.0 25.0
5492031.06293 10000.0 0.1
9527439024.39 10000.0 10.0
1.18185148854e+014 10000.0 25.0
549203106.293 100000.0 0.1
952743902439.0 100000.0 10.0
1.18185148854e+016 100000.0 25.0
Head for given time and distance with shut-off after 1e7 s
using list input and output
```

```
time [s], distance [m], pressure [Pa]
[[100.0, 10.0, 28461.4939044692],
[100.0, 100.0, 0.064903171252350678],
[100.0, 5000.0, 0.0], [100.0, 10000.0, 0.0],
[100.0, 100000.0, 0.0],
[10000.0, 10.0, 98854.956722670351],
[10000.0, 100.0, 28461.4939044692],
[10000.0, 5000.0, 1.6603581610482317e-107],
[10000.0, 10000.0, 0.0], [10000.0, 100000.0, 0.0],
[1000000.0, 10.0, 170740.48590584149],
[1000000.0, 100.0, 98854.956722670351],
[1000000.0, 5000.0, 388.99910214021651],
[1000000.0, 10000.0, 0.064903171252350678],
[1000000.0, 100000.0, 0.0],
[1000000000.0, 10.0, 156.91689630621113],
[1000000000.0, 100.0, 156.9167401753075],
[1000000000.0, 5000.0, 156.52312259559403],
[1000000000.0, 10000.0, 155.34771582833491],
[1000000000.0, 100000.0, 57.436417790474025]]
```

References

[1] C. V. Theis, The relation between lowering of the piezometric surface and the rate and duration of discharge of a well using ground water storage, Trans. Am. Geophys. Union 16th anual meeting (Pt. 2) (1935) 519–524.