HantushJacob
a Python class implementing the Hantush-Jacob solution for the pressure perturbation due to injection into an aquifer with leaky layers
Version 0.2

Karl W. Bandilla

May 7, 2010

© Karl Bandilla 2010

KWBandilla@gmail.com

1 Introduction

HantushJacob is a class programmed in the scripting language Python. The purpose of the class is to compute the pressure perturbation caused by injecting a fluid into an aquifer which is sandwiched between either one or two leaky layers. The analytic solution implemented in this class is based on the classic solution by Hantush and Jacob [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

HantushJacob 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

HantushJacob_0_2.py and *ExampleHantushJacob_0_2.py* can be freely downloaded from code.google.com/p/camelotpy. In order to use *HantushJacob_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 *HantushJacob_0_1.py*. Alternatively, Python can be installed from www.python.org. In this case the package *SciPy* needs to be installed separately. *HantushJacob_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:

- Q: volumetric injection rate [L^3/T] (float). Positive into the aquifer.
- S: storativity of the aquifer (volume of water that a unit decline of head releases from storage in a vertical prism of the aquifer of unit cross section) [-] (float).
- T: hydraulic transmissivity of the aquifer [L^2/T] (float).
- B: transmissivity-leakance ratio [L] (float).
- *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, because the given head or pressure is above the steady-state head or pressure. 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 *HantushJacob* class consists of several functions and references three generic modules (*SciPy*, *SciPy.special*, and *math*). The functions are:

- *SetRequiredProperties(Q, S, T, B, 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 transmissivity-leakance ratio (B), 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), 'B' transmissivity-leakance ratio (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.

- *FindB(aquifer_transmissivity, aquitard_conductivities, aquitard_thicknesses)*: takes the aquifer transmissivity (*aquifer_transmissivity*) [L^2/T] (float), the list of leaky layer conductivities (*aquitard_conductivities*) [L/T] (list of floats; 1 entry for one leaky layer; 2 entries for 2 leaky layers), and the list of leaky layer thicknesses (*aquitard_thicknesses*) [L] (list of floats; 1 entry for one leaky layer; 2 entries for 2 leaky layers). Returns the transmissivity-leakance ratio [L] (float) upon completion.

- *EvalWellFunction(u, rOverB)*: evaluates the Hantush-Jacob well function using the dimensionless parameter ($u = \frac{S\ radius^2}{4\ T\ time}$) [-] (float) and the dimensionless radius ($rOverB = \frac{radius}{B}$) [-] (float). Returns the value of the Hantush-Jacob well function [-] (float) upon return.

- *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 less than the steady state head or pressure for the given radius, 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 *HantushJacob* 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 *ExampleHantushJacob_0_2.py*, and the module *HantushJacob_0_2.py* should either be in same directory as *ExampleHantushJacob_0_2.py* or be located within the Python search path.

```
import HantushJacob_0_2 as hj

# model properties
Ha = 500.0 #m, aquifer thickness
Ht = 200.0 #m, thickness of top leaky layer
Hb = 100.0 #m, thickness of bottom leaky layer
Q = 0.1 #m^3/s, injection rate
Ka = 1e-5 #m/s, aquifer hydraulic conductivity
Kt = 4e-12 #m2, hydraulic conductivity of top leaky layer
Kb = 4e-11 #m2, hydraulic conductivity of bottom leaky layer
```

```
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, 50.0]
#Specify distance from well [m]
radius = [10.0, 100.0, 5000.0, 10000.0, 100000.0]

#initialize the model
model = hj.HantushJacob()

#compute B
B = model.FindB(T, [Kt, Kb], [Ht, Hb])
model.SetRequiredProperties(Q, S, T, B)

#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
model.SetProperty('WarnNoRadius', ['i', -2])
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
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 pressures 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 'time [s], distance [m], pressure [Pa]'
print 'using list input and output'
print model.GetHeadForRadiusTime(radius, times)
```

Running the *ExampleHantushJacob_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 2.90127356074
100.0 100.0 6.62611854635e-006
100.0 5000.0 0.0
100.0 10000.0 0.0
100.0 100000.0 0.0
10000.0 10.0 10.076954554
10000.0 100.0 2.9012711738
10000.0 5000.0 0.0
10000.0 10000.0 0.0
10000.0 100000.0 0.0
1000000.0 10.0 17.4044044565
1000000.0 100.0 10.0766261142
1000000.0 5000.0 0.039646707034
1000000.0 10000.0 6.61464148934e-006
1000000.0 100000.0 0.0
1000000000.0 10.0 28.0812905603
1000000000.0 100.0 20.7519408094
1000000000.0 5000.0 8.30883144429
1000000000.0 10000.0 6.12559021054
1000000000.0 100000.0 0.303176258159
Contour radius for given head and time
time [s], distance [m], head [m]
100.0 42.6710440408 0.1
100.0 1.02449739103 10.0
```

```
100.0 -2.0 50.0
10000.0 426.710311957 0.1
10000.0 10.2449633517 10.0
10000.0 -2.0 50.0
1000000.0 4266.97301097 0.1
1000000.0 102.439055279 10.0
1000000.0 0.000357052473558 50.0
1000000000.0 130810.668585 0.1
1000000000.0 2933.89937696 10.0
1000000000.0 0.0102202920357 50.0
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
pressure not reached 10.0 50.0
549.203106293 100.0 0.1
952934.561984 100.0 10.0
pressure not reached 100.0 50.0
1373124.655 5000.0 0.1
6594131223.21 5000.0 10.0
pressure not reached 5000.0 50.0
5493871.58625 10000.0 0.1
pressure not reached 10000.0 10.0
pressure not reached 10000.0 50.0
568886461.64 100000.0 0.1
pressure not reached 100000.0 10.0
pressure not reached 100000.0 50.0
Head for given time and distance with shut-off after 1e7 s
time [s], distance [m], pressure [Pa]
[[100.0, 10.0, 28461.493630842499],
[100.0, 100.0, 0.065002222939686605],
[100.0, 5000.0, 0.0], [100.0, 10000.0, 0.0],
[100.0, 100000.0, 0.0],
[10000.0, 10.0, 98854.924175177017],
[10000.0, 100.0, 28461.470215022018],
[10000.0, 5000.0, 0.0], [10000.0, 10000.0, 0.0],
[10000.0, 100000.0, 0.0],
 [1000000.0, 10.0, 170737.20771832051],
```

```
[1000000.0, 100.0, 98851.702180134176],
[1000000.0, 5000.0, 388.93419600385994],
[1000000.0, 10000.0, 0.064889633010393993],
[1000000.0, 100000.0, 0.0],
[1000000000.0, 10.0, 127.32823563698912],
[1000000000.0, 100.0, 127.32810894629802],
[1000000000.0, 5000.0, 127.00871236658713],
[1000000000.0, 10000.0, 126.05494170882594],
[1000000000.0, 100000.0, 46.605974143322783]]
```

References

[1] M. S. Hantush, C. E. Jacob, Nonsteady radial flow in an infinite leaky aquifer, Trans. Am. Geophys. Union 36th anual meeting (Pt. 1) (1955) 95–100.