nz-stream-depletion Documentation

Release 1.0.0

Mike Kittridge

Oct 26, 2021

SECTIONS

This python package contains a class and various functions for calculating stream depletion from pumping wells. The stream depletion functions used in this package are derived from groundwater analytical solutions. Most of these were originally developed to support New Zealand regional councils in their water resource management. The stream depletion methods in this python package were transcribed from Excel VBA macros written by Bruce Hunt. Matt Smith from Environment Canterbury created a complementary excel spreadsheet that has built upon Bruce Hunt's spreadsheet to be more user-friendly, and he also provided valuable input to the development of this package. This package was inspired by this other stream depletion python package by Wilco Terink.

The GitHub repository is found here.

CHAPTER

ONE

INSTALLATION

Install via pip:

```
pip install nz-stream-depletion
```

Or conda:

```
conda install -c mullenkamp nz-stream-depletion
```

1.1 Requirements

The main dependencies are numpy and pandas.

CHAPTER

TWO

STREAM DEPLETION METHODS

2.1 Background

All of these stream depletion methods are derivations from the fundamental groundwater principles. They have been specifically derived to estimate the ratio of the stream water compared to the surrounding groundwater during pumping of a nearby well. In all of the following conceptual models, they represent simplified representations of the groundwater system to ensure that the input requirements are minimised. The complexity of the conceptual groundwater model increases with the input requirements. This page describes the methods in a sequence from the most simple to the most complex. The names used in the python package will be labeled in **bold** for reference.

A very good review of the various stream depletion solutions can be found in Bruce Hunt's Review of Stream Depletion Solutions, Behavior, and Calculations. Many of the following diagrams come from this paper.

2.2 Input aquifer (and stream) parameters

All methods have aquifer and stream input parameter requirements. This section will list all of the input parameter names used in the python package with a description of what they mean (glossary):

```
      sep_distance : int
    The separation distance from the pumped well to the stream.
pump_aq_trans : int
    The pumped (confined) aquifer transmissivity (m2/day).
pump_aq_s : float
    The storage coefficient of the pumped aquifer.
upper_aq_trans: int
    The surficial aquifer transmissivity (m2/day).
upper_aq_s : float
    The storage coefficient of the surficial aquifer.
lower_aq_trans: int
    The confined aquifer transmissivity (m2/day).
lower_aq_s : float
    The storage coefficient (specific storage) of the confined aquifer.
aqt_k : int
    The aquitard hydraulic conductivity (m/day).
aqt_s : float
    The aquitard storage coefficient.
aqt_thick : int
    The aquitard vertical thickness (m).
stream_k : int
```

(continues on next page)

(continued from previous page)

```
    Streambed hydraulic conductivity (m/day).
stream_thick : int
    The streambed vertical thickness (m).
stream_width : int
    The streambed width (m).
```

2.3 Theis 1941

The Theis 1941 solution labeled **theis_1941** in the python package represents a straight river fully penetrating a homogeneous, isotropic aquifer of semi-infinite extent, which can be either confined or unconfined.

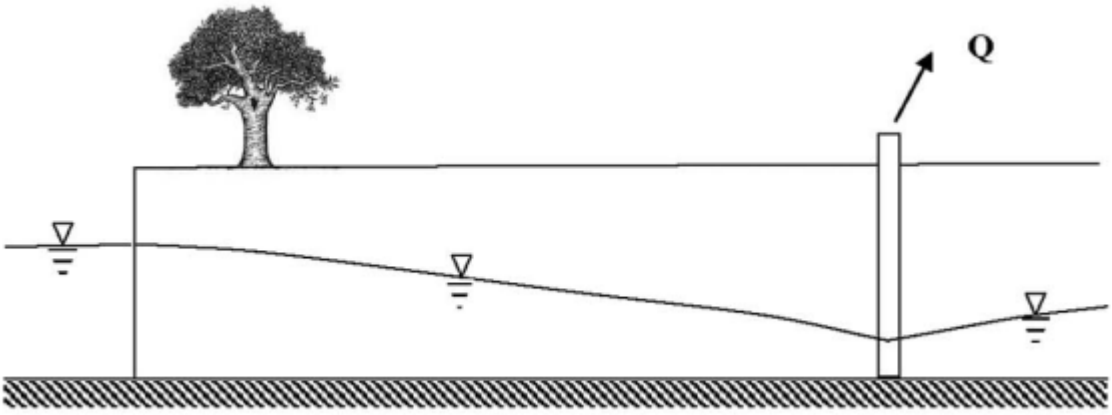

Fig. 1: The conceptual model diagram of Theis 1941.

2.3.1 input parameters

- sep_distance
- pump_aq_trans
- pump_aq_s

2.4 Hunt 1999

The Hunt 1999 solution labeled **hunt_1999** in the python package represents a stream that partially penetrates an aquifer extending to infinity in all horizontal directions. Where the aquifer is again homogeneous and isotropic with a semi-infinite extent.

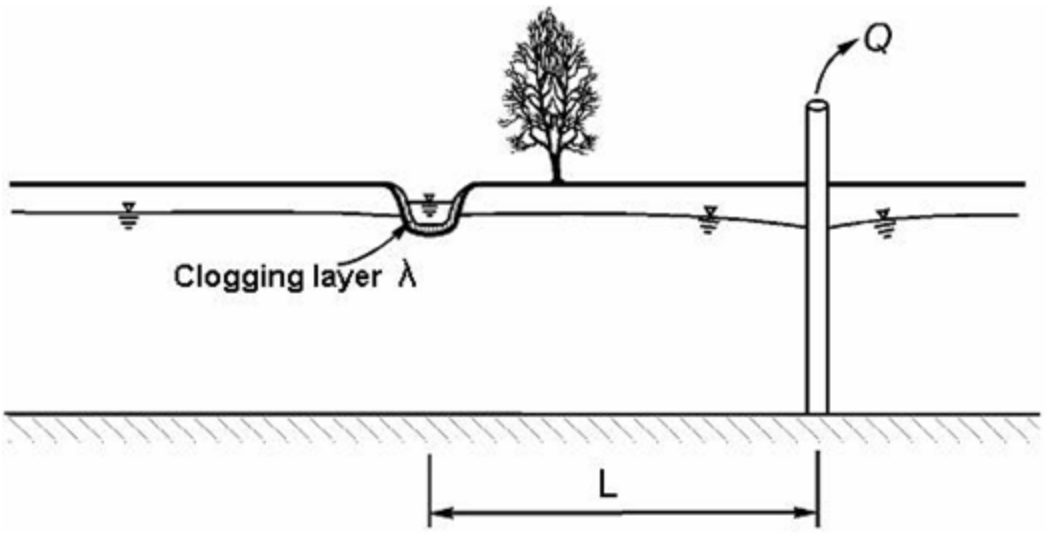

Fig. 2: The conceptual model diagram of Hunt 1999.

2.4.1 input parameters

- sep_distance
- pump_aq_trans
- pump_aq_s
- stream_k
- stream_thick
- stream_width

2.5 Hunt 2003

The Hunt 2003 solution labeled **hunt_2003** in the python package represents a similar conceptual model as the Hunt 1999 except that the pumped aquifer underlays an aquitard containing a free surface.

2.5.1 input parameters

- sep_distance
- pump_aq_trans
- pump_aq_s
- stream_k
- stream_thick
- stream_width
- aqt_k
- aqt_thick
- aqt_s

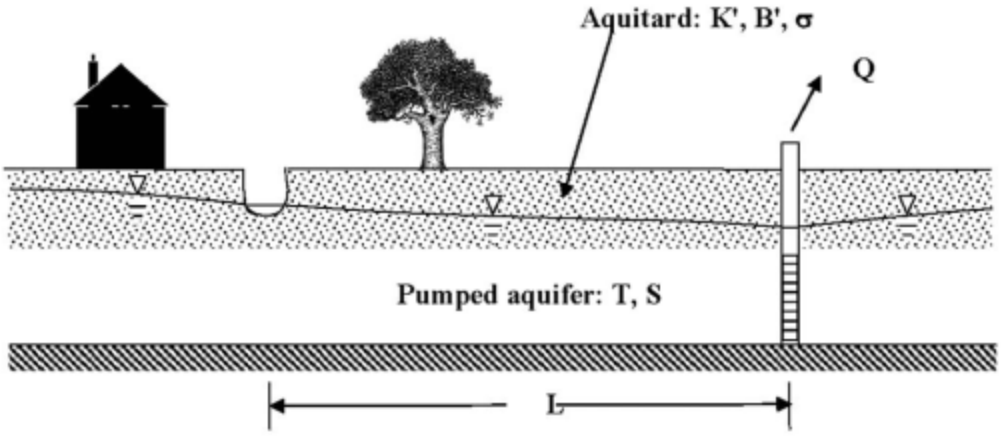

Fig. 3: The conceptual model diagram of Hunt 2003.

2.6 Hunt 2009

The Hunt 2009 solution labeled **hunt_2009** in the python package represents a similar conceptual model as the Hunt 1999 except that the pumped aquifer overlays an aquitard and a second unpumped confined aquifer.

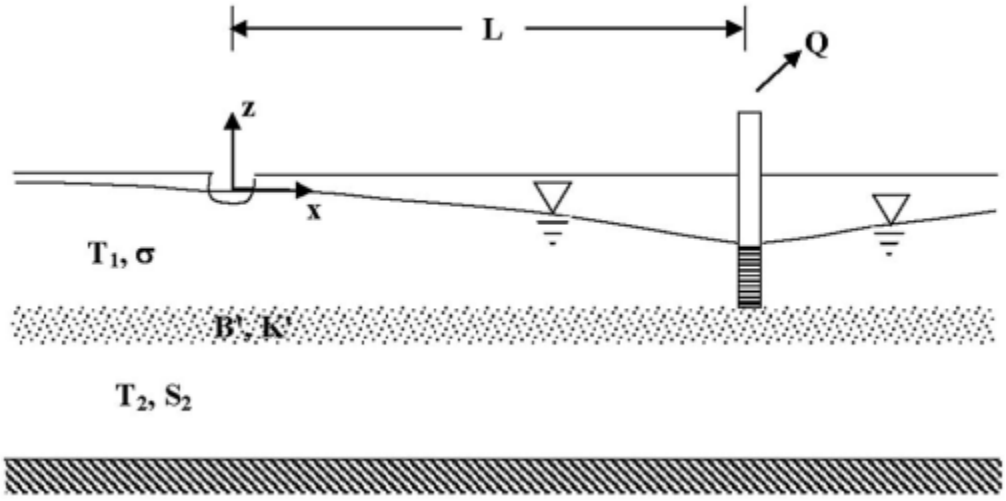

Fig. 4: The conceptual model diagram of Hunt 2009.

2.6.1 input parameters

- sep_distance
- pump_aq_trans
- pump_aq_s
- stream_k
- stream_thick
- stream_width
- aqt_k
- aqt_thick
- lower_aq_trans
- lower_aq_s

2.7 Ward and Lough 2011

The Ward and Lough 2011 solution labeled **ward_lough_2011** in the python package represents a similar conceptual model as the Hunt 2009 except that the pumped aquifer is now the lower confined aquifer.

2.7.1 input parameters

- sep_distance
- pump_aq_trans
- pump_aq_s
- stream_k
- stream_thick
- stream_width
- aqt_k
- aqt_thick
- upper_aq_trans
- upper_aq_s

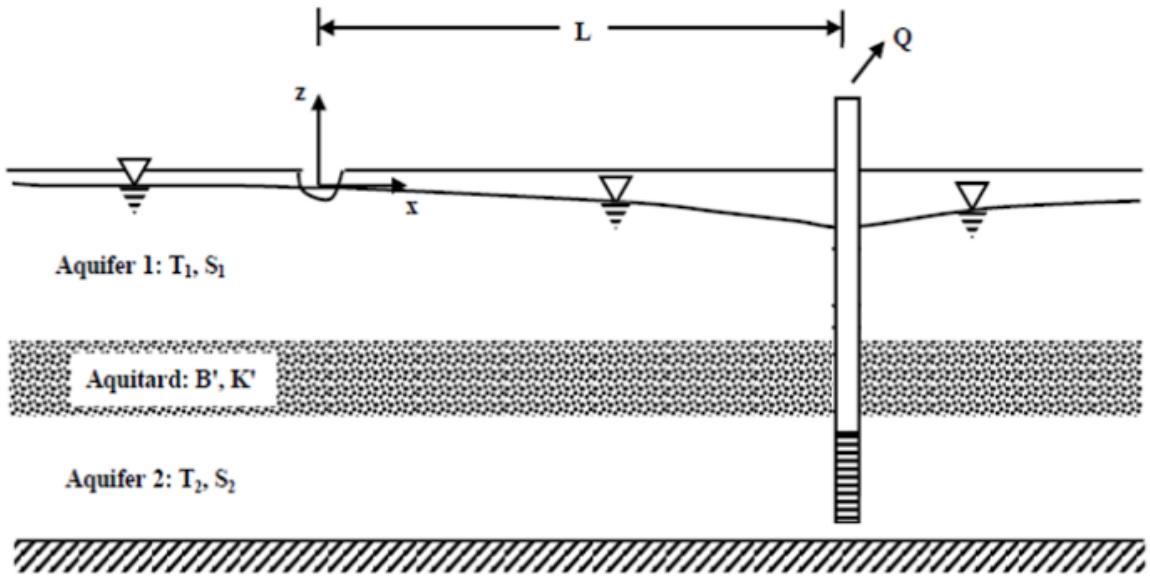

Fig. 5: The conceptual model diagram of Ward and Lough 2011.

CHAPTER

THREE

HOW TO USE NZ-STREAM-DEPLETION

This section will describe how to use the nz-stream-depletion package.

3.1 The SD class

The SD class in the nz-stream-depletion package provides access to all of the stream depletion methods withouthaving to specify each function. You just feed in the appropriate input parameters and the SD will use the most appropriate method given the input parameters.

First we need to import the package and define some of those input aquifer parameters.

```
from nz_stream_depletion import SD

# Pumped aquifer
pump_aq_trans = 1000 # m/d
pump_aq_s = 0.1

# Well
pump_q = 50 # l/s
sep_distance = 500 # m

time1 = 6 # days
time2 = 7 # days
time1 = 30 # days
n_days = 150 # days

# streambed
stream_k = 1000 # m/d
stream_thick = 1 # m
stream_width = 1 # m
stream_cond = stream_k * stream_thick / stream_width

# aquitard
aqt_k = 0.1 # m/d
aqt_thick = 10 # m
aqt_s = 0.1

# upper aquifer
upper_aq_trans = 500
upper_aq_s = 0.05
```

Then we need to initialize the SD class. Once the SD class is initialized, we can take a look at the all_methods attribute to see all of the emthods and the parameter requirements.

```
In [1]: sd = SD()

In [2]: print(sd.all_methods)
{'theis_1941': ['pump_aq_trans', 'pump_aq_s', 'sep_distance'], 'hunt_1999': ['pump_aq_
↪trans', 'pump_aq_s', 'stream_k', 'stream_thick', 'stream_width', 'sep_distance'],
↪'hunt_2003': ['pump_aq_trans', 'pump_aq_s', 'aqt_k', 'aqt_thick', 'aqt_s', 'stream_k',
↪'stream_thick', 'stream_width', 'sep_distance'], 'hunt_2009': ['pump_aq_trans', 'pump_
↪aq_s', 'lower_aq_trans', 'lower_aq_s', 'aqt_k', 'aqt_thick', 'stream_k', 'stream_thick
↪', 'stream_width', 'sep_distance'], 'ward_lough_2011': ['pump_aq_trans', 'pump_aq_s',
↪'upper_aq_trans', 'upper_aq_s', 'aqt_k', 'aqt_thick', 'stream_k', 'stream_thick',
↪'stream_width', 'sep_distance']}
```

Then we need to load the input aquifer parameters. The available methods attribute tells us what methods are available given the input parameters.

```
In [3]: sd = SD()

In [4]: available = sd.load_aquifer_data(pump_aq_trans=pump_aq_trans, pump_aq_s=pump_aq_
↪s, sep_distance=sep_distance)

In [5]: print(available)
['theis_1941']
```

Once the input parameters have been loaded, you can calculate the stream depletion ratios using either sd.calc_sd_ratio for a specific number of pumping days (n_days) or sd.calc_sd_ratios for all of the ratios up to the pumping days. It can be helpful in certain coding circumstances to put the input parameters into a dictionary before passing them to the SD class.

```
In [6]: params2 = {'pump_aq_trans': pump_aq_trans, 'pump_aq_s': pump_aq_s, 'sep_distance
↪': sep_distance, 'stream_k': stream_k, 'stream_thick': stream_thick, 'stream_width':␣
↪stream_width}

In [7]: sd = SD()

In [8]: available = sd.load_aquifer_data(**params2)

In [9]: sd_ratio = sd.calc_sd_ratio(7)

In [10]: sd_ratios = sd.calc_sd_ratios(7)

In [11]: print(available)
['theis_1941', 'hunt_1999']

In [12]: print(sd_ratio)
0.17979760932708438

In [13]: print(sd_ratios)
[8.170694148276362e-05, 0.012081873210892132, 0.041004577820555124, 0.07656642910427265,␣
↪0.11288495663350082, 0.14756681656258458, 0.17979760932708438]
```

The last bit of functionality allows you to take a time series of extraction (pumping) data and determine the amount that is stream depleting over the entire record. The application of these stream depletion methods on variable pumping

rates over time uses the superposition principle.

```
In [14]: params2 = {'pump_aq_trans': pump_aq_trans, 'pump_aq_s': pump_aq_s, 'sep_distance
↪': sep_distance, 'stream_k': stream_k, 'stream_thick': stream_thick, 'stream_width':␣
↪stream_width}

In [15]: extract_csv = 'https://raw.githubusercontent.com/mullenkamp/nz-stream-depletion/
↪main/nz_stream_depletion/data/sample_flow.csv'

In [16]: extraction = pd.read_csv(extract_csv, index_col='time', parse_dates=True, infer_
↪datetime_format=True, dayfirst=True).flow

In [17]: sd = SD()

In [18]: available = sd.load_aquifer_data(**params2)

In [19]: sd_rates = sd.calc_sd_extraction(extraction)

In [20]: print(sd_rates)
time
2010-10-06    0.000037
2010-10-07    0.005496
2010-10-08    0.018537
2010-10-09    0.034576
2010-10-10    0.051558
                ...
2020-06-20    0.174635
2020-06-21    0.180846
2020-06-22    0.191781
2020-06-23    0.201990
2020-06-24    0.209487
Freq: D, Length: 3550, dtype: float64
```

CHAPTER

FOUR

PACKAGE REFERENCES

4.1 Base class

class nz_stream_depletion.**SD**

4.2 Input aquifer parameters

SD.**load_aquifer_data**(*sep_distance: int*, *pump_aq_trans: int*, *pump_aq_s: float*, *upper_aq_trans: Optional[int] = None*, *upper_aq_s: Optional[float] = None*, *lower_aq_trans: Optional[int] = None*, *lower_aq_s: Optional[float] = None*, *aqt_k: Optional[int] = None*, *aqt_thick: Optional[int] = None*, *aqt_s: Optional[float] = None*, *stream_k: Optional[int] = None*, *stream_thick: Optional[int] = None*, *stream_width: Optional[int] = None*)

This method is where the physical properties of the aquifer(s) and the stream are assigned and processed. The minimum required data includes the sep_distance, pump_aq_trans, and the pump_aq_s. It will determine which stream depletion methods are available based on the input data.

Parameters

- **sep_distance** (*int*) – The separation distance from the pumped well to the stream.
- **pump_aq_trans** (*int*) – The pumped (confined) aquifer transmissivity (m2/day).
- **pump_aq_s** (*float*) – The storage coefficient of the pumped aquifer.
- **upper_aq_trans** (*int*) – The surficial aquifer transmissivity (m2/day).
- **upper_aq_s** (*float*) – The storage coefficient of the surficial aquifer.
- **lower_aq_trans** (*int*) – The confined aquifer transmissivity (m2/day).
- **lower_aq_s** (*float*) – The storage coefficient (specific storage) of the confined aquifer.
- **aqt_k** (*int*) – The aquitard hydraulic conductivity (m/day).
- **aqt_s** (*float*) – The aquitard storage coefficient.
- **aqt_thick** (*int*) – The aquitard vertical thickness (m).
- **stream_k** (*int*) – Streambed hydraulic conductivity (m/day).
- **stream_thick** (*int*) – The streambed vertical thickness (m).
- **stream_width** (*int*) – The streambed width (m).

Returns of available methods based on the input data

Return type list

4.3 stream depletion calcs

SD.**calc_sd_ratio**(*n_days: int*, *method: Optional[str] = None*)

Calculate the stream depletion ratio for a specific number of pumping days for a specific method. If no method is provided, it will choose the highest ranking method based on the available methods.

Parameters

- **n_days** (*int*) – The number of pumping days.
- **method** (*str or None*) – The stream depletion method to use. It must be one of the available methods based on the input data. None will select the highest ranking method based on the available methods.

Returns

Return type float

SD.**calc_sd_ratios**(*n_days: int*, *method: Optional[str] = None*)

Calculate the stream depletion ratios for all pumping days up to the n_days for a specific method. If no method is provided, it will choose the highest ranking method based on the available methods.

Parameters

- **n_days** (*int*) – The number of pumping days.
- **method** (*str or None*) – The stream depletion method to use. It must be one of the available methods based on the input data. None will select the highest ranking method based on the available methods.

Returns

Return type list

SD.**calc_sd_extraction**(*extraction: pandas.core.series.Series*, *method: Optional[str] = None*)

Calculate the stream depleting extraction for all days in the extraction time series for a specific method. If no method is provided, it will choose the highest ranking method based on the available methods.

Parameters

- **extraction** (*pd.Series*) – The extraction time series. It must contain a pd.DatetimeIndex. If the series is irregular, it will make it regular and pad the missing times with zero. Currently, only daily data is allowed as input.
- **method** (*str or None*) – The stream depletion method to use. It must be one of the available methods based on the input data. None will select the highest ranking method based on the available methods.

Returns

Return type list

4.4 API Pages

—

CHAPTER

FIVE

LICENSE AND TERMS OF USAGE

This package is licensed under the terms of the Apache License Version 2.0.

INDEX

C

L

S