TimML Documentation

Release 5.0.0

M. Bakker

Oct 06, 2017

CONTENTS

TimML is a computer program for the modeling of steady-state multi-layer flow with analytic elements TimML may be applied to an arbitrary number of layers and arbitrary sequence of aquifers and leaky layers. The Dupuit approximation is applied to aquifer layers, while flow in leaky layers is approximated as vertical. The head, flow, and leakage between aquifer layers may be computed analytically at any point in the aquifer system. The design of TimML is object-oriented and has been kept simple and flexible. New analytic elements may be added to the code without making any changes in the existing part of the code. TimML is coded in Python. Behind the scenes, use is made of FORTRAN extensions to improve performance.

CHAPTER

ONE

INSTALLATION

TimML is written for Python 3.

pip install instructions will follwo soon.

CHAPTER

TWO

MAIN APPROXIMATIONS

CHAPTER

THREE

LIST OF AVAILABLE ELEMENTS

- Well
  - Discharge-specified well
  - Head-specified well
  - Multi-aquifer well. Well is screened in multiple layers and only total discharge is specified.
- Line-sink
  - Head-specified line-sink
  - String of head-specified line-sinks

3.1 Models

3.1.1 Multi-Aquifer Model

class timml.model.**ModelMaq**(*kaq=1, z=[1, 0], c=[], npor=0.3, top='conf', hstar=None*)

ModelMaq Class to create a multi-aquifer model object

Parameters

- **kaq** (*float, array or list*) – hydraulic conductivity of each aquifer from the top down if float, hydraulic conductivity is the same in all aquifers
- **z** (*array or list*) – elevation tops and bottoms of the aquifers from the top down leaky layers may have zero thickness if top='conf': length is 2 * number of aquifers if top='semi': length is 2 * number of aquifers + 1 as top of leaky layer on top of systems needs to be specified
- **c** (*float, array or list*) – resistance of leaky layers from the top down if float, resistance is the same for all leaky layers if top='conf': length is number of aquifers - 1 if top='semi': length is number of aquifers
- **npor** (*float, array or list*) – porosity of all aquifers and leaky layers from the top down if float, porosity is the same for all layers if top='conf': length is 2 * number of aquifers - 1 if top='semi': length is 2 * number of aquifers
- **top** (*string, 'conf' or 'semi' (default is 'conf')*) – indicating whether the top is confined ('conf') or semi-confined ('semi')
- **hstar** (*float or None (default is None)*) – head value above semi-confining top, only read if top='semi'

Examples

```
>>> ml = ModelMaq(kaq=[10, 20], z=[20, 12, 10, 0], c=1000)
```

disvec(*x*, *y*, *aq=None*)
Discharge vector at *x*, *y*

Returns **qxqy** – first row is Qx in each aquifer layer, second row is Qy

Return type array size (2, naq)

head(*x*, *y*, *layers=None*, *aq=None*)
Head at *x*, *y*

Returns **h** – head in all *layers* (if not *None*), or all layers of aquifer (otherwise)

Return type array length *naq* or *len(layers)*

headalongline(*x*, *y*, *layers=None*)
Returns head[Nlayers,len(x)] Assumes same number of layers for each x and y layers may be None or list of layers for which head is computed

headgrid(*xg*, *yg*, *layers=None*, *printrow=False*)
Returns h[Nlayers,ny,nx]. If layers is None, all layers are returned

headgrid2(*x1*, *x2*, *nx*, *y1*, *y2*, *ny*, *layers=None*, *printrow=False*)
Returns h[Nlayers,ny,nx]. If layers is None, all layers are returned

remove_element(*e*)
Remove element *e* from model

solve(*printmat=0*, *sendback=0*, *silent=False*)
Compute solution

3.1.2 Three-dimensional flow model

Similar to LPF in MODFLOW

class timml.model.**Model3D**(*kaq=1, z=[1, 0], kzoverkh=1, npor=0.3, top='conf', topres=0, topthick=0, hstar=0*)
Model3D Class to create a multi-layer model object consisting of many aquifer layers. The resistance between the layers is computed from the vertical hydraulic conductivity of the layers.

Parameters

- **kaq** (*float, array or list*) – hydraulic conductivity of each layer from the top down if float, hydraulic conductivity is the same in all aquifers
- **z** (*array or list*) – elevation of top of system followed by bottoms of all layers from the top down bottom of layer is automatically equal to top of layer below it if top='conf': length is number of layers + 1 if top='semi': length is number of layers + 2 as top of leaky layer on top of systems needs to be specified
- **kzoverkh** (*float*) – vertical anisotropy ratio vertical k divided by horizontal k if float, value is the same for all layers length is number of layers
- **npor** (*float, array or list*) – porosity of all aquifer layers from the top down if float, porosity is the same for all layers if top='conf': length is number of layers if top='semi': length is number of layers + 1

- **top** (*string, 'conf' or 'semi' (default is 'conf')*) – indicating whether the top is confined ('conf') or semi-confined ('semi')
- **topres** (*float*) – resistance of top semi-confining layer, only read if top='semi'
- **topthick** (*float*) – thickness of top semi-confining layer, only read if top='semi'
- **hstar** (*float or None (default is None)*) – head value above semi-confining top, only read if top='semi'

Examples

```
>>> ml = Model3D(kaq=10, z=np.arange(20, -1, -2), kzoverkh=0.1)
```

disvec(*x*, *y*, *aq=None*)
Discharge vector at *x*, *y*

Returns **qxqy** – first row is Qx in each aquifer layer, second row is Qy

Return type array size (2, naq)

head(*x*, *y*, *layers=None*, *aq=None*)
Head at *x*, *y*

Returns **h** – head in all *layers* (if not *None*), or all layers of aquifer (otherwise)

Return type array length *naq* or *len(layers)*

headalongline(*x*, *y*, *layers=None*)
Returns head[Nlayers,len(x)] Assumes same number of layers for each x and y layers may be None or list of layers for which head is computed

headgrid(*xg*, *yg*, *layers=None*, *printrow=False*)
Returns h[Nlayers,ny,nx]. If layers is None, all layers are returned

headgrid2(*x1*, *x2*, *nx*, *y1*, *y2*, *ny*, *layers=None*, *printrow=False*)
Returns h[Nlayers,ny,nx]. If layers is None, all layers are returned

remove_element(*e*)
Remove element *e* from model

solve(*printmat=0*, *sendback=0*, *silent=False*)
Compute solution

3.1.3 Model

class timml.model.**Model**(*kaq*, *c*, *z*, *npor*, *ltype*)
Model Class to create a model object consisting of an arbitrary sequence of aquifer layers and leaky layers. Use ModelMaq for regular sequence of aquifer and leaky layers. Use Model3D for multi-layer model of single aquifer

Parameters

- **kaq** (*array*) – hydraulic conductivity of each aquifer from the top down
- **z** (*array*) – elevation tops and bottoms of all layers layers may have zero thickness
- **c** (*array*) – resistance between two consecutive aquifers if ltype[0]='a': length is number of aquifers - 1 if ltype[0]='l': length is number of aquifers
- **npor** (*array*) – porosity of all layers from the top down

- **ltype** (*array of characters*) – array indicating for each layer whether it is 'a' aquifer layer 'l' leaky layer

Examples

```
>>> from timml import *
>>> ml = Model(kaq=array([10, 20, 10]), c=array([200, 2000]),
↪z=array([20, 15, 10, 8, 0]), npor=0.3 * ones(4),              ltype=array(['a',
↪'a', 'l', 'a']))
```

disvec(*x*, *y*, *aq=None*)
Discharge vector at *x*, *y*

Returns **qxqy** – first row is Qx in each aquifer layer, second row is Qy

Return type array size (2, naq)

head(*x*, *y*, *layers=None*, *aq=None*)
Head at *x*, *y*

Returns **h** – head in all *layers* (if not *None*), or all layers of aquifer (otherwise)

Return type array length *naq* or *len(layers)*

headalongline(*x*, *y*, *layers=None*)
Returns head[Nlayers,len(x)] Assumes same number of layers for each x and y layers may be None or list of layers for which head is computed

headgrid(*xg*, *yg*, *layers=None*, *printrow=False*)
Returns h[Nlayers,ny,nx]. If layers is None, all layers are returned

headgrid2(*x1*, *x2*, *nx*, *y1*, *y2*, *ny*, *layers=None*, *printrow=False*)
Returns h[Nlayers,ny,nx]. If layers is None, all layers are returned

remove_element(*e*)
Remove element *e* from model

solve(*printmat=0*, *sendback=0*, *silent=False*)
Compute solution

3.2 Inhomogeneities

3.3 Elements

3.3.1 Wells

Well

class timml.well.**Well**(*model*, *xw=0*, *yw=0*, *Qw=100.0*, *rw=0.1*, *res=0.0*, *layers=0*, *label=None*)
Well Class to create a well with a specified discharge. The well may be screened in multiple layers. The resistance of the screen may be specified. The head is computed such that the discharge Q_i in layer i is computed as

$$Q_i = 2\pi r_w (h_i - h_w)/c$$

where c is the resistance of the well screen and h_w is the head inside the well. The total discharge is distributed over the screens such that h_w is the same in each screened layer.

Parameters

- **model** (*Model object*) – model to which the element is added
- **xw** (*float*) – x-coordinate of the well
- **yw** (*float*) – y-coordinate of the well
- **Qw** (*float*) – total discharge of the well
- **rw** (*float*) – radius of the well
- **res** (*float*) – resistance of the well screen
- **layers** (*int, array or list*) – layer (int) or layers (list or array) where well is screened
- **label** (*string or None (default: None)*) – label of the well

Examples

```
>>> ml = Model3D(kaq=10, z=np.arange(20, -1, -2), kzoverkh=0.1)
>>> Well(ml, 100, 200, 1000, layers=[0, 1, 2, 3])
```

capzone(*nt=10*, *zstart=None*, *hstepmax=10*, *vstepfrac=0.2*, *tmax=None*, *nstepmax=100*, *silent='.'*)
Compute a capture zone

Parameters

- **nt** (*int*) – number of path lines
- **zstart** (*scalar*) – starting elevation of the path lines
- **hstepmax** (*scalar*) – maximum step in horizontal space
- **vstepfrac** (*float*) – maximum fraction of aquifer layer thickness during one step
- **tmax** (*scalar*) – maximum time
- **nstepmax** (*scalar (int)*) – maximum number of steps
- **silent** (*boolean or string*) – True (no messages), False (all messages), or '.' (print dot for each path line)

Returns xyzt

Return type list of arrays of x, y, z, and t values

discharge()
The discharge in each layer

Returns Discharge in each screen with zeros for layers that are not screened

Return type array (length number of layers)

headinside()
The head inside the well

Returns Head inside the well for each screen

Return type array (length number of screens)

plotcapzone(*nt=10, zstart=None, hstepmax=20, vstepfrac=0.2, tmax=365, nstepmax=100, silent='.', color=None, orientation='hor', win=[-1e+30, 1e+30, -1e+30, 1e+30], newfig=False, figsize=None*)

Plot a capture zone

Parameters

- **nt** (*int*) – number of path lines
- **zstart** (*scalar*) – starting elevation of the path lines
- **hstepmax** (*scalar*) – maximum step in horizontal space
- **vstepfrac** (*float*) – maximum fraction of aquifer layer thickness during one step
- **tmax** (*scalar*) – maximum time
- **nstepmax** (*scalar (int)*) – maximum number of steps
- **silent** (*boolean or string*) – True (no messages), False (all messages), or '.' (print dot for each path line)
- **color** (*color*) –
- **orientation** (*string*) – 'hor' for horizontal, 'ver' for vertical, or 'both' for both
- **win** (*array_like (length 4)*) – [xmin, xmax, ymin, ymax]
- **newfig** (*boolean (default False)*) – boolean indicating if new figure should be created
- **figsize** (*tuple of integers, optional, default: None*) – width, height in inches.

HeadWell

class timml.well.**HeadWell**(*model, xw=0, yw=0, hw=10, rw=0.1, res=0, layers=0, label=None*)

HeadWell Class to create a well with a specified head inside the well. The well may be screened in multiple layers. The resistance of the screen may be specified. The head is computed such that the discharge Q_i in layer i is computed as

$$Q_i = 2\pi r_w (h_i - h_w)/c$$

where c is the resistance of the well screen and h_w is the head inside the well.

Parameters

- **model** (*Model object*) – model to which the element is added
- **xw** (*float*) – x-coordinate of the well
- **yw** (*float*) – y-coordinate of the well
- **hw** (*float*) – head inside the well
- **rw** (*float*) – radius of the well
- **res** (*float*) – resistance of the well screen
- **layers** (*int, array or list*) – layer (int) or layers (list or array) where well is screened
- **label** (*string (default: None)*) – label of the well

capzone(*nt=10*, *zstart=None*, *hstepmax=10*, *vstepfrac=0.2*, *tmax=None*, *nstepmax=100*, *silent='.'*)
Compute a capture zone

Parameters

- **nt** (*int*) – number of path lines
- **zstart** (*scalar*) – starting elevation of the path lines
- **hstepmax** (*scalar*) – maximum step in horizontal space
- **vstepfrac** (*float*) – maximum fraction of aquifer layer thickness during one step
- **tmax** (*scalar*) – maximum time
- **nstepmax** (*scalar (int)*) – maximum number of steps
- **silent** (*boolean or string*) – True (no messages), False (all messages), or '.' (print dot for each path line)

Returns xyzt

Return type list of arrays of x, y, z, and t values

discharge()
The discharge in each layer

Returns Discharge in each screen with zeros for layers that are not screened

Return type array (length number of layers)

headinside()
The head inside the well

Returns Head inside the well for each screen

Return type array (length number of screens)

plotcapzone(*nt=10, zstart=None, hstepmax=20, vstepfrac=0.2, tmax=365, nstepmax=100, silent='.', color=None, orientation='hor', win=[-1e+30, 1e+30, -1e+30, 1e+30], newfig=False, figsize=None*)
Plot a capture zone

Parameters

- **nt** (*int*) – number of path lines
- **zstart** (*scalar*) – starting elevation of the path lines
- **hstepmax** (*scalar*) – maximum step in horizontal space
- **vstepfrac** (*float*) – maximum fraction of aquifer layer thickness during one step
- **tmax** (*scalar*) – maximum time
- **nstepmax** (*scalar (int)*) – maximum number of steps
- **silent** (*boolean or string*) – True (no messages), False (all messages), or '.' (print dot for each path line)
- **color** (*color*) –
- **orientation** (*string*) – 'hor' for horizontal, 'ver' for vertical, or 'both' for both
- **win** (*array_like (length 4)*) – [xmin, xmax, ymin, ymax]
- **newfig** (*boolean (default False)*) – boolean indicating if new figure should be created

- **figsize** (*tuple of integers, optional, default: None*) – width, height in inches.

3.3.2 Line-sinks

There are many line-sinks

Head-specified line-sink

class timml.linesink.**HeadLineSink**(*model*, *x1=-1*, *y1=0*, *x2=1*, *y2=0*, *hls=1.0*, *res=0*, *wh=1*, *layers=0*, *label=None*, *addtomodel=True*)

HeadLineSinkString

class timml.linesink.**HeadLineSinkString**(*model, xy=[(-1, 0), (1, 0)], hls=0, res=0, wh=1, order=0, layers=0, label=None, name='HeadLineSinkString'*)

HeadLineSinkString Class to create a string of head-specified line-sinks which may optionally have a width and resistance :param model: Model to which the element is added :type model: Model object :param xy: elevation tops and bottoms of the aquifers from the top down

leaky layers may have zero thickness if top='conf': length is 2 * number of aquifers if top='semi': length is 2 * number of aquifers + 1 as top of leaky layer on top of systems needs to be specified

Parameters

- **hls** (*scalar, array or list*) – head along string if scalar: head is the same everywhere along the string if list or array of length 2: head at beginning and end of string if list or array with same length as xy: heads at nodes, which may contain nans, except for first and last point
- **res** (*scalar (default is 0)*) – resistance of line-sink
- **wh** (*scalar or str*) –
- **order** (*int (default is 0)*) – order of all line-sinks in string
- **layers** (*scalar, list or array*) – layer(s) in which element is placed if scalar: element is placed in this layer if list or array: element is placed in all these layers
- **label** (*str or None*) –
- **-----** –

Examples

```
>>> ml = ModelMaq(kaq=[10, 20], z=[20, 12, 10, 0], c=1000)
>>> HeadLineSinkString(ml, xy=[(-1, 0), (1, 0), (0, 1)], hls=5)
```

Line-sink ditch

Specified total discharge with unknown but uniform head

class timml.linesink.**LineSinkDitch**(*model*, *x1=-1*, *y1=0*, *x2=1*, *y2=0*, *Qls=1*, *res=0*, *wh=1*, *order=0*, *layers=0*, *label=None*, *addtomodel=True*)

3.3.3 Line-doublets

Impermeable wall

class timml.linedoublet.**ImpLineDoublet**(*model*, *x1=-1*, *y1=0*, *x2=1*, *y2=0*, *order=0*, *layers=0*, *label=None*, *addtomodel=True*)

Leaky wall

class timml.linedoublet.**LeakyLineDoublet**(*model*, *x1=-1*, *y1=0*, *x2=1*, *y2=0*, *res=0*, *order=0*, *layers=0*, *label=None*, *addtomodel=True*)

3.3.4 Uniform Flow

class timml.uflow.**Uflow**(*model*, *slope*, *angle*, *name='Uflow'*, *label=None*)

3.3.5 Area-sinks

Circular Area-Sink

class timml.circareasink.**CircAreaSink**(*model*, *xc=0*, *yc=0*, *R=1*, *N=0.001*, *layer=0*, *name='CircAreasink'*, *label=None*)

3.4 Utilities

INDEX