User Guide for Spowtd v0.4.0

Alex Cobb

This is the user guide for Spowtd, which implements the scalar parameterization of water table dynamics described in Cobb et al. [1] and Cobb and Harvey [2].

1 The steps of scalar parameterization

Scalar parameterization involves these essential steps:

1. Load water level, precipitation and evapotranspiration data;
2. Identify dry intervals and storm intervals;
3. Match intervals of rising water levels to rainstorms;
4. Construct a master rising curve;
5. Construct a master recession curve;
6. Fit a preliminary specific yield function to the master rising curve;
7. Jointly fit a specific yield and a conductivity (equivalently, transmissivity) function to the master rising and recession curves.

2 The spowtd script

The `spowtd` script provides a command-line interface to perform calculations with Spowtd.

2.1 Dependencies

Running the script requires Python 3 and the Python packages Matplotlib, Numpy, and Pytz.

2.2 Using the script

The `spowtd` script has these subcommands (typically run in this order):

- `spowtd load`: Load water level, precipitation and evapotranspiration data.
- `spowtd classify`: Classify data into storm and interstorm intervals.
- `spowtd set-zeta-grid`: Set up water level grid for master curves.
- `spowtd recession`: Assemble recession curve.
- `spowtd rise`: Assemble rise curve.
- `spowtd plot`: Plot data.
- `spowtd set-curvature`: Set site curvature.
- `spowtd simulate`: Simulate data rise curve, recession curve, or rising and receding intervals.
- `spowtd pestfiles`: Generate input files for calibration with PEST.

The first step is to load the precipitation, evapotranspiration and water level data. The input text files must be in an UTF-8-compatible encoding (ASCII is fine). The time zone is stored with the dataset and will be used in plots (all times are stored internally as UNIX timestamps). For example, to load data into a new dataset file called `ekolongouma.sqlite3`:

```
spowtd load ekolongouma.sqlite3 \
  -vvv \
  --precipitation src/precipitation_Ekolongouma.txt \
  --evapotranspiration src/evapotranspiration_Ekolongouma.txt \
  --water-level src/waterlevel_Ekolongouma.txt \
  --timezone Africa/Lagos
```

The verbosity flags (`-vvv`) are not required; they cause the script to report more on what is being done.

Next, classify the water level and precipitation time series into storm and interstorm intervals based on thresholds for rainfall intensity and rates of increase in water level. For example, this command classifies intervals with precipitation of at least 4 mm / h as storms, and intervals in which the water level is increasing at a rate of least 8 mm / h as storm response.

```
spowtd classify ekolongouma.sqlite3 \
  -vvv \
  --storm-rain-threshold-mm-h 4.0 \
  --rising-jump-threshold-mm-h 8.0
```

At this stage the classification can be plotted. A basic interactive plot showing the classified water level and precipitation time series can be produced with:

```
spowtd plot time-series ekolongouma.sqlite3
```

An additional panel showing evapotranspiration is plotted if the `-e` or `--plot-evapotranspiration` flag is passed. The parts of the water level time series marked as interstorms are on a light red background, and the parts of the water level time series marked as storm response are on a light green background. The parts of the precipitation time series marked as storms are on a light blue background. You can pan in the plot with the right mouse button and zoom with a left mouse button, or use the magnifying glass to zoom in. You can revert to earlier zoom and pan values with the arrow buttons.

Adding `-f` or `--flags` highlights the parts of the water level time series that have been classified as storm response and interstorms, and the parts of the precipitation time series

```
spowtd plot time-series ekolongouma.sqlite3  -f
```

The rising intervals are highlighted in blue, intervals with rising intervals that could not be matched to rain storms are highlighted in magenta, and rain storms are highlighted in red.

The next step is to establish a uniform grid for water levels. This grid is used when storm and interstorm intervals are assembled into rising and recession curves.

```
spowtd set-zeta-grid -vvv ekolongouma.sqlite3
```

The next two steps assemble the recession and rise curves:

```
spowtd recession -vvv ekolongouma.sqlite3
```

```
spowtd rise -vvv ekolongouma.sqlite3
```

The recession and rise curves are now assembled, and can be plotted.

```
spowtd plot recession ekolongouma.sqlite3
```

```
spowtd plot rise ekolongouma.sqlite3
```

These plots can be interacted with in the same way: left mouse button to pan, right mouse button to zoom, disk icon to save.

3 Parameterization

Parameters are provided to `spowtd` in YAML format.

Currently two types of parameter sets are supported: (1) Cubic spline for specific yield, piecewise linear for the logarithm of conductivity; and (2) The PEATCLSM parameterization.

The spline parameterizations look like this:

```
specific_yield:
  type: spline
  zeta_knots_mm:
    - -291.7
    - -183.1
    - -15.74
    - 10.65
    - 38.78
    - 168.3
  sy_knots:  # Specific yield, dimensionless
    - 0.1358
    - 0.1671
    - 0.2541
    - 0.2907
    - 0.2892
    - 0.6857
transmissivity:
  type: spline
  zeta_knots_mm:
    - -291.7
    - -5.167
    - 168.3
    - 1000
```

```
  K_knots_km_d:  # Conductivity, km/d
    - 5.356e-3
    - 1.002
    - 6577.0
    - 8.430e+3
  minimum_transmissivity_m2_d: 7.442  # Minimum transmissivity, m2/d
```

and the PEATCLSM parameterizations look like this:

```
specific_yield:
  type: peatclsm
  sd: 0.162  # standard deviation of microtopographic distribution, m
  theta_s: 0.88  # saturated moisture content, m^3/m^3
  b: 7.4  # shape parameter, dimensionless
  psi_s: -0.024  # air entry pressure, m
transmissivity:
  type: peatclsm
  Ksmacz0: 7.3  # m/s
  alpha: 3  # dimensionless
  zeta_max_cm: 5.0
```

(the text following each parameter, after the #, is a comment and invisible to spowtd).

The specific yield and transmissivity curves can be plotted with

```
  spowtd plot WHAT parameters.yml
```

where `WHAT` is one of `specific-yield`, `conductivity` or `transmissivity` and `parameters.yml` is a YAML file containing hydraulic parameters.

The plotting commands `plot rise`, `plot recession` and `plot time-series` support a parameter `-p`, `--parameters`; if a YAML file containing hydraulic parameters is passed to one of these commands, the corresponding plot (rising curve, recession curve, rising and receding intervals) is simulated using those parameters.

The simulated curves and corresponding data can be obtained as text using `spowtd simulate WHAT data.sqlite3 parameters.yml` where `WHAT` is `rise`, `recession`, or `intervals`. These commands write simulated data, water level data, and / or residuals to an output file (standard output by default) as delimited text. For example,

```
spowtd simulate rise ekolongouma.sqlite3 parameters.yml
```

reads data from `ekolongouma.sqlite3` and parameters from the file `parameters.yml` and writes the assembled and simulated rise curves to standard output.

To simulate (or plot) recession requires setting the large-scale curvature of the site. The command

```
  spowtd set-curvature ekolongouma.sqlite3 1.0
```

sets the site curvature to 1 m/km/km, whereafter

```
spowtd simulate recession ekolongouma.sqlite3 parameters.yml
```

simulates the water table recession.

4 Calibration with PEST

The simulation scripts make it possible to calibrate the specific yield and transmissivity functions against rise and recession of the water level using the PEST software package and tools for model-independent parameter estimation and uncertainty analysis. It should also be possible to calibrate using PEST++, which is designed to have the same text-based interface, by following a similar procedure.

PEST is a highly configurable set of tools. One of its strengths is that it is possible to start with a fairly simple approach and incorporate more sophisticated functionality as it is needed. As an introduction, we illustrate calibration of specific yield parameters against the rise curve.

For a calibration with PEST, you need to create five text files:

1. A PEST control file (`.pst`), which configures how PEST will perform the calibration (including identifying the other files used during calibration);
2. A parameter template file (`.tpl`), into which PEST will substitute parameter values in a format that can be read by Spowtd;
3. An output template file, or PEST "instruction file" (`.ins`), which teaches PEST how to extract "observations" from `spowtd simulate` output;
4. A vector of initial parameters (`.par`) to start the calibration; and
5. A script to execute the rise simulation.

The first step is to create the PEST control file (`.pst`) following the PEST documentation [3]. For a PEATCLSM parameterization, the control file will describe the four PEATCLSM parameters for specific yield (`sd`, `theta_s`, `b`, and `psi_s`), each with their own parameter group. The control file must also include an "observation data" section with a single line giving mean dynamic storage for each water level in the rise curve. The "model command line" section must provide the command line needed to run the script to generate the rise curve. The script itself can be, for example, a bash script that calls `spowtd simulate rise`. Finally, the "model input/output" section specifies the path to the parameter template file, the path to the the parameter file that PEST will create by substituting parameter values into the template, the path to the instruction file (`.ins`) that PEST uses to interpret the simulation output, and the path to the output file created by a single run of the simulation script.

The parameter template file (`.tpl`) and the parameter vector file (`.par`) are created by replacing values in a Spowtd YAML parameter file by placeholders, as described in the PEST documentation. In the case of calibration of PEATCLSM specific yield parameters against a rise curve, the template file might look like this:

```
ptf @
specific_yield:
  type: peatclsm
  sd: @sd                        @
  theta_s: @theta_s                    @
  b: @b                         @
  psi_s: @psi_s                      @
transmissivity:
  type: peatclsm
  Ksmacz0: 7.3  # m/s
  alpha: 3  # dimensionless
  zeta_max_cm: 5.0
```

and an initial parameter vector file might look like this:

```
double point
          sd    0.162                        1.000000         0.000000
    theta_s     0.88                         1.000000         0.000000
           b    7.4                          1.000000         0.000000
       psi_s    -0.024                       1.000000         0.000000
```

To verify the format of a template `rise_pars.yml.tpl` and initial parameters `rise_init.par`, use the PEST `tempchek` command:

```
tempchek rise_pars.yml.tpl rise_pars.yml rise_init.par
```

This command should exit without errors and produce a valid parameter file at `rise_pars.yml`.

The parameter file can then be verified by running your script. Your script might, for example, contain the command

```
spowtd simulate rise ekolongouma.sqlite3 rise_pars.yml -o rise_observations.yml
    --observations
```

which generates simulated dynamic storage values (without water levels or measured dynamic storage values) in `rise_observations.yml`; in PEST, simulated output values are referred to as "observations."

The resulting output file can then be checked against a PEST instruction file (`.ins`) that you create for extracting observation data, which might be called `rise_observations.ins`, using the PEST command `inschek`:

```
inschek rise_observations.ins rise_observations.yml
```

To then ensure that the correct initial parameters are used in the calibration, substitute these into the control file using `parrep`

```
parrep rise_init.par rise_calibration.in.pst rise_calibration.pst
```

To then calibrate specific yield parameters against the rise curve (alone) using the PEST control file `rise_calibration`, call:

```
pestchek rise_calibration &&
(pest rise_calibration.pst ;
 tempchek rise_pars.yml.tpl rise_opt.yml rise_calibration.par)
```

These commands check the PEST control file, perform the calibration, and then substitute the calibrated parameter values from `rise_calibration.par` into `rise_opt.yml`.

You can then examine the fit by plotting the rise curve with the calibrated parameters:

```
spowtd plot rise ekolongouma.sqlite3 --parameters rise_opt.yml
```

4.1 Generating PEST input files with Spowtd

As a convenience, Spowtd can generate input files for calibration with PEST, either against the rise curve (`spowtd pestfiles rise`) or against both rise and recession curves (`spowtd pestfiles curves`). The arguments to both subcommands are the same. Taking calibration against the rise curve as an example, a template file can be created with

```
spowtd pestfiles rise ekolongouma.sqlite3 parameters.yml tpl \\
  -o rise_parameters.yml.tpl
```

An instruction file can similarly be created with

```
spowtd pestfiles rise ekolongouma.sqlite3 parameters.yml ins \\
  -o ekolongouma_rise_observations.ins
```

and a control file can be created with

```
spowtd pestfiles rise ekolongouma.sqlite3 parameters.yml pst \\
  -o ekolongouma_rise_calibration.in.pst
```

The template and instruction files can be used as-is. The generated PEST control file will require substitution of valid starting parameters and bounds, substitution of paths to input files and the invocation for simualtion, adjustment of PEST control parameters, etc.

References

[1] Alexander R. Cobb, Alison May Hoyt, Laure Gandois, Jangarun Eri, René Dommain, Kamariah Abu Salim, Fuu Ming Kai, Nur Salihah Haji Su'ut, and Charles F. Harvey. How temporal patterns in rainfall determine the geomorphology and carbon fluxes of tropical peatlands. *Proceedings of the National Academy of Sciences of the United States of America*, 114: E5187–E5196, 2017. doi: 10.1073/pnas.1701090114.

[2] Alexander R. Cobb and Charles F. Harvey. Scalar simulation and parameterization of water table dynamics in tropical peatlands. *Water Resources Research*, 55(11):9351–9377, 2019. doi: 10.1029/2019wr025411.

[3] John Doherty. *PEST: Model-Independent Parameter Estimation User Manual*. Watermark Numerical Computing, 5th edition, 2010.