FREEWAT

Free and Open Source Software Tools for Water Resource Management

EU HORIZON 2020 Project

# FREE and Open Source Software Tools for WATer Resource Management

## FREEWAT User Manual – Volume 0

### Reference Manual

Version 1.0
September 30th, 2017

This page has been intentionally left blank

# FREEWAT User Manual - Volume 0

## Reference Manual

**Version 1.0**
**September 30th, 2017**

By I. Borsi(1), L. Foglia(2), M. Cannata(3), E. Vázquez-Suñé(4), S. Mehl(5), G. De Filippis(6), R. Criollo(4), M. Ghetta(6), M. Cardoso(3), V. Velasco(4), J. Neumann(3), A. Toegl(2), A. Serrano(4), C. Riera(4), and R. Rossetto(6)

(1) TEA Sistemi SpA, Pisa (IT)
(2) Institut für Angewandte Geowissenschaften – TU Darmstadt, Darmstadt (DE)
(3) Istituto di Scienze della Terra – SUPSI, Canobbio (CH)
(4) Institute of Environmental Assessment and Water Research (IDÆA), Barcelona (ES)
(5) Department of Civil Engineering – California State University, Chico (CA)
(6) Istituto di Scienze della Vita – Scuola Superiore Sant'Anna, Pisa (IT)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 642224

This project document reflects only the authors' views and the European Union is not liable for any use that may be made of the information contained therein.

FREEWAT Development has received funding from the following projects:

1. Hydrological part has been developed starting from a former project, named SID&GRID, funded by Regione Toscana through EU POR-FSE 2007-2013 (sidgrid.isti.cnr.it)

2. Porting of SID&GRID under QGis has been performed through funds provided by Regione Toscana to Scuola Superiore S. Anna - Project Evoluzione del sistema open source SID&GRID di elaborazione dei dati geografici vettoriali e raster per il porting negli ambienti QGis e Spatialite in uso presso la Regione Toscana (CIG: ZA50E4058A)

3. Saturated zone solute transport simulation capability has been developed within the EU FP7-ENV-2013-WATER-INNO-DEMO MARSOL. MARSOL project receives funding from the European Union's Seventh Framework Programme for Research, Technological Development and Demonstration under grant agreement n. 619120 (www.marsol.eu)

4. Latest Version of FREEWAT is under development within EU H2020 project FREEWAT - Free and Open Source Software Tools for Water Resource Management. FREEWAT project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement n. 642224 (www.freewat.eu)

Suggested citation:

I. Borsi, L. Foglia, M. Cannata, E. Vázquez-Suñé, S. Mehl, G. De Filippis, R. Criollo, M. Ghetta, M. Cardoso, V. Velasco, J. Neumann, A. Toegl, A. Serrano, C. Riera, and R. Rossetto. FREEWAT User Manual, Volume 0 – Reference Manual, version 1.0, September 30th, 2017.

License:

# Contents

# Abstract

This manual contains a theoretical background to the modules included in the FREEWAT platform, developed within the HORIZON 2020 FREEWAT project (www.freewat.eu). For each module, an overview on the conceptualization of the tool is presented, followed by its implementation within the FREEWAT platform. The Reader is referred to the cited references to get information on particular aspects of the codes included in each module, while a guidance to apply each tool is documented in FREEWAT User's Manuals (Vol 1 – 6).

Some illustrative examples implementing different FREEWAT capabilities are reported as well, mainly taking such experiences from the 14 case studies run during the H2020 FREEWAT project. That Section is particularly useful for the User approaching FREEWAT for the first time, and willing to know examples of successful application of the modeling platform.

Finally, Appendixes A to H report the Programmer Documentation for each module of the FREEWAT plugin.

# 1 Introduction

FREEWAT is conceived as a composite plugin for the well-known GIS open source desktop software QGIS (http://qgis.org). The selected reference version of QGIS is the latest LTR (Long Term Release), namely QGIS 2.14: even if this release will be maintained as the reference one, it is worth mentioning that any test performed so far with subsequent versions (e.g. 2.16 and 2.18) worked without experiencing any problem. In Fig. 1.1 the QGIS bar with the FREEWAT menu is reported, as example.

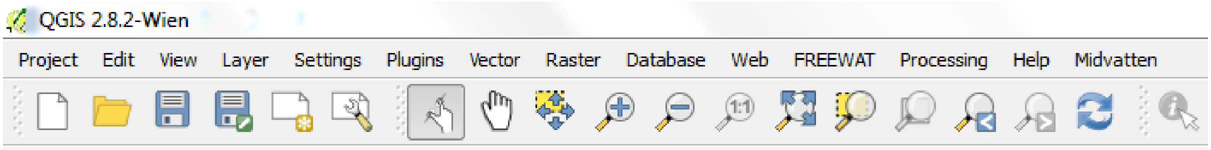


*Figure 1.1 - The QGIS bar with the FREEWAT plugin menu included.*

As composite plugin, FREEWAT is designed as a modular ensemble of different tools: some of them can be used independently, while some modules require the preliminary execution of other tools. In this framework, the following tool classifications can be defined:

- Tools for the analysis, interpretation and visualization of hydrogeological and hydrochemical data and quality issues, also focusing on advanced time series analysis, embedded in **akvaGIS** module.

- Simulation of models related to the hydrological cycle and water resources management: flow models, transport models, crop growth models, management and optimization models (also related to irrigation management and rural issues).

- Tools to perform model calibration, sensitivity analysis and uncertainty quantifications.

- Additional tools for general GIS operations to prepare input data, and post-processing functionalities (module **OAT – Observation and Analysis Tool**).

The structure of the FREEWAT Menu reflects as much as possible this classification, as shown in the screen shot reported in Fig. 1.2.

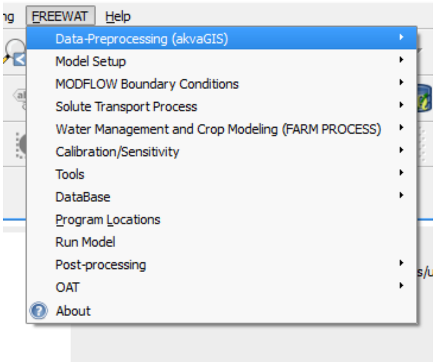


*Figure 1.2 – The FREEWAT menu expanded.*

The diagram reported in Fig. 1.3 shows how these different modules are interconnected, taking as reference a standard modeling procedure.

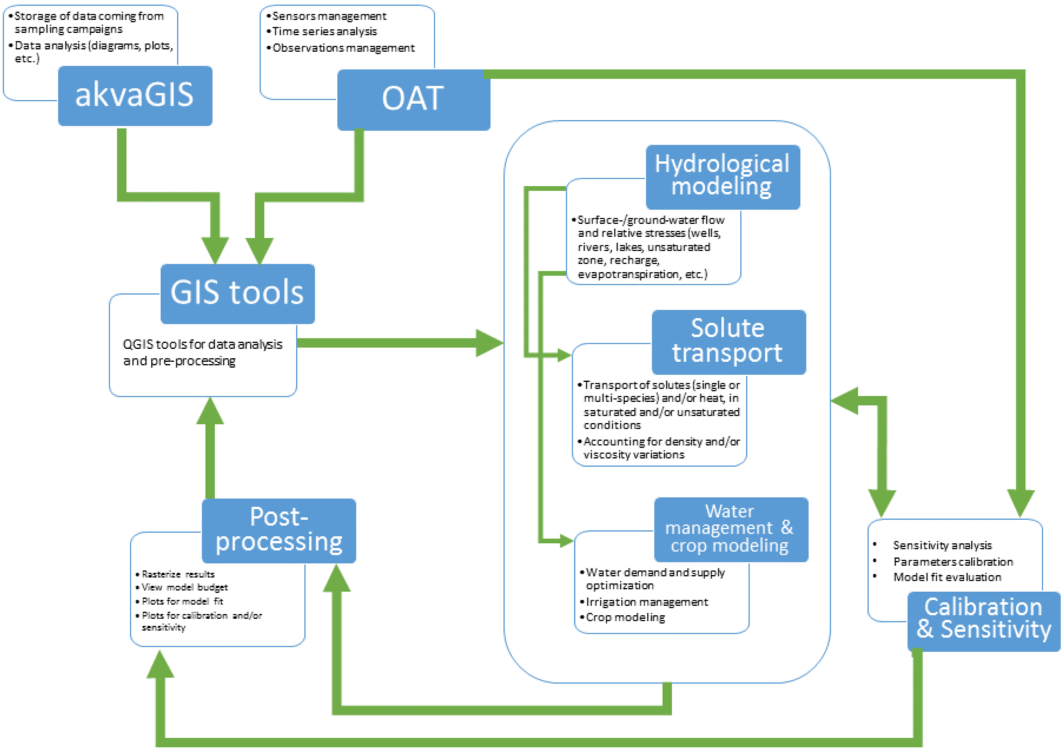

•Storage of data coming from sampling campaigns
•Data analysis (diagrams, plots, etc.)
akvaGIS
•Sensors management
•Time series analysis
•Observations management
OAT
GIS tools
QGIS tools for data analysis and pre-processing
Hydrological modeling
•Surface-/ground-water flow and relative stresses (wells, rivers, lakes, unsaturated zone, recharge, evapotranspiration, etc.)
Solute transport
•Transport of solutes (single or multi-species) and/or heat, in saturated and/or unsaturated conditions
•Accounting for density and/or viscosity variations
Water management & crop modeling
•Water demand and supply optimization
•Irrigation management
•Crop modeling
Post-processing
•Rasterize results
•View model budget
•Plots for model fit
•Plots for calibration and/or sensitivity
Sensitivity analysis
Parameters calibration
Model fit evaluation
Calibration & Sensitivity

*Figure 1.3 – Interconnection among FREEWAT tools and modules.*

FREEWAT architecture is based on the integration of different software tools (the so called FREEWAT pillars): SQLITE relational database manager, external (free and open source) codes like MODFLOW and MODFLOW-related programs as well as codes specifically developed for the FREEWAT. The way of interconnecting such tools is done via Python programming language, with extensive use of the Python library FloPy. A schematic representation of FREEWAT pillars and their interconnection is showed in Fig. 1.4.

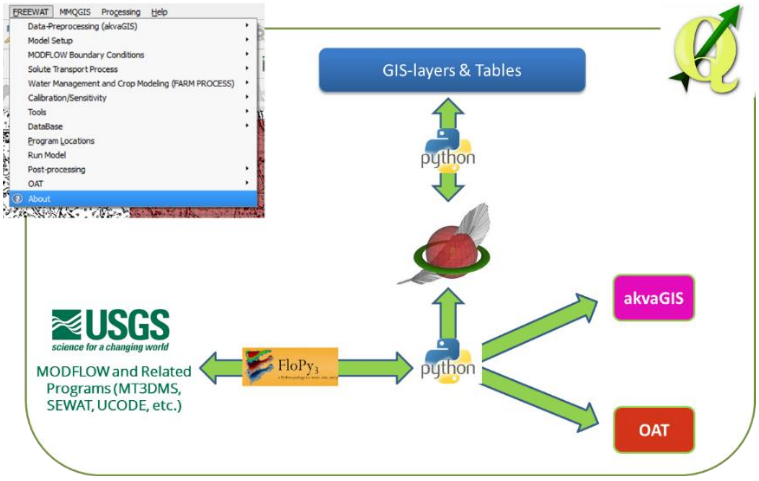


*Figure 1.4 – FREEWAT pillars.*

# 2 Groundwater flow modeling

The main component of FREEWAT plugin consists of a suite of modelling tools for performing groundwater flow, and related processes. The hydrological model implemented in FREEWAT allows simulating the entire hydrological cycle, provided that climate data, like rainfall and temperature, are available. However, it is also possible to focus only on selected parts of the model.

The simulated processes cited in this section are:

- groundwater flow in the saturated zone, including interaction with surface water bodies (e.g., rivers, lakes, drains);
- vertical flow through the unsaturated zone and beneath surface water streams.

## 2.1 Conceptualization of groundwater modeling in FREEWAT

The modelling framework is based on the worldwide known 3D finite difference groundwater flow MODFLOW and related codes (McDonald and Harbaugh, 1988), by integrating primarily the MODFLOW-2005 version (Harbaugh, 2005). MODFLOW is a physically-based, spatially distributed code developed by USGS, which simulates the groundwater flow dynamics in the saturated and unsaturated zones, both in confined and unconfined aquifers with constant or variable thickness and transmissivity values, in steady-state or transient conditions. In MODFLOW applications, space is discretized in the form of a regular grid, while time is discretised in stress periods (i.e., time intervals during which hydrologic stresses do not change), which in turn may be divided in smaller time steps. MODFLOW source code, written in FORTRAN, is open, well documented, freely available on the web at https://water.usgs.gov/ogw/modflow/, and it has become global standard for groundwater modelling applications. MODFLOW requires text input files using well-defined format and also a specific file structure.

MODFLOW has a wide range of different Packages which allow to simulate several processes. The current FREEWAT plugin includes the majority of these Packages which are useful for the simulation of the main hydrological and hydrogeological processes.

Hereinafter, a list of the MODFLOW Packages (Harbaugh, 2005), grouped in categories, is provided:

- Basic Packages:
  - Basic (BAS)
  - Discretization (DIS)
  - Layer Property Flow (LPF)
- Hydrogeological processes:
  - specified-head boundaries
    - Time-Variant Specified Head (CHD)
  - specified-flux boundaries
    - Recharge (RCH)

    - Well (WEL)
  - head-dependent flux
    - Unsaturated Zone Flow (UZF)
    - River (RIV)
    - Lake (LAK)
    - Drain (DRN)
    - General-Head Boundary (GHB)
    - Evapotranspiration (EVT)
    - Multi-Node Well (MNW)
    - Stream Flow Routing (SFR2)

Specific conceptualization for each package is here omitted. The Reader is referred to the huge amount of documentation materials available to understand MODFLOW conceptualization (among these, we mention the FREEWAT MODFLOW Lectures available in FREEWAT website, http://www.freewat.eu ).

As exception, some additional information on LAK package is reported below, since this tool presents some specific aspects not included in other packages.

## 2.2 Conceptualization of Lake Package

The MODFLOW Lak7 package is implemented as one of the boundary condition options in the FREEWAT modelling environment. The Lak7 package allows for the simulation of hydraulic interaction between a lake and groundwater so that the effects of the changes in the water level of one of the two water bodies is calculated on the other. In this function the lake package differs greatly from previous methods used to simulate surface water bodies. Previously, these were the constant head package, river package (Harbaugh, 2005), the reservoir package (Fenske, Leake & Prudic, 1996) or the “High K” method outlined by Merrit & Konikow (2000).

The Lake package exists as part of the core hydrological modelling module as a MODFLOW boundary condition. Its interface allows the user to specify lake variables needed by MODFLOW such as lake leakance, or precipitation among others, and use the GIS tools to specify the location of the lakes, and the feature attribute tables where necessary.

The lake package finds its use when the dynamic interaction between surface water and groundwater is the subject of investigation. Simplified approaches, often used in the past take into account either surface water or ground water alone. Surface water models simplify interactions with the groundwater and simple MODFLOW models simplify the interaction with the surface water. The LAK package was first developed by Cheng & Anderson (1993) to simulate three-dimensional steady-state and transient ground-water flow with fluctuating lake levels, taking into account, precipitation, evaporation, streamflow and groundwater flow. Council (1997) demonstrated a real world example used to predict lake level decline in response to a proposed underground mine. The USGS (Krohelski et. al., 2002) also uses the lake package to investigate the causes of increased lake stages in three shallow seepage lakes and to propose a solution pumping scheme to lower water levels based on a calibrated model. The dynamics of lakes was investigated by Virdi and Lee (2009) in their investigation of Lake Starr with the unsaturated zone and stream packages, a lake which fluctuates more than 13 feet during a 10 years period, as well as expanding between 96 to 148 acres between dry and wet years. The permeable sand hills were simulated using the UZF1 package, allowing additional model cells to be exposed to recharge and rainfall as the lake recedes. This approach is particularly suited to simulating surface-water bodies such as wetlands and playa lakes (Virdi & Lee, 2009). They were able to show the effects of climate extremes and pumping as stresses on the flow from and to lakes and its surface area (Verdi et. al., 2012).

The lake package has also been used to assess the quantitative interaction of lakes with the environment and groundwater within the scope of a contamination of lake sediments.

### 2.2.1 Conceptualisation of the lake package in FREEWAT

Within FREEWAT the Lake package is part of the module on hydrological modelling, as seen in Fig. 1.1. The lake package can be classified as a MODFLOW boundary condition, and so falls within this category, although it is significantly more complex that some other boundary conditions, e.g. constant flow or constant head. The lake package calculates a separate water budget for those cells identified as lake removing them from the solution of the groundwater equation and setting them as inactive. Active groundwater cells adjacent to the lakes exchange water at a rate determined by the relative calculated head in the aquifer and stage in the lake, hydraulic conductivities of the aquifer and lakebed materials, and the area of the lakes. As result the lake stage is recalculated every time step.

MODFLOW, which is the core of FREEWAT, is a finite-element model. Thus its elements are defined by a discretization grid in three dimensions simulated by consecutive raster grid cells with depth information. The Lake package is not an exception and the lake volume is calculated by the 3D cells identified as lake. As a consequence of the finite-difference approach, the lake stage is related to lake volume directly through the volume of the model grid cells marked as lake cells, and their water level. Lake stage is crucial for determining the groundwater flow from and to the lake. As the lake cells are inactive for the groundwater flow equation, the bottom of the lake is identical to the bottom elevation of the grid cell. Adjusting the height of a cell is approximate to adjusting lake bathymetry (Fig. 2.1). Additionally, the lake can dry completely and re-wet during transient simulations if the lake stage drops below this cell bottom.

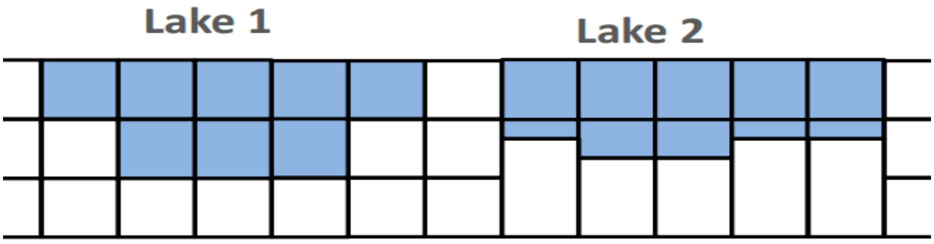


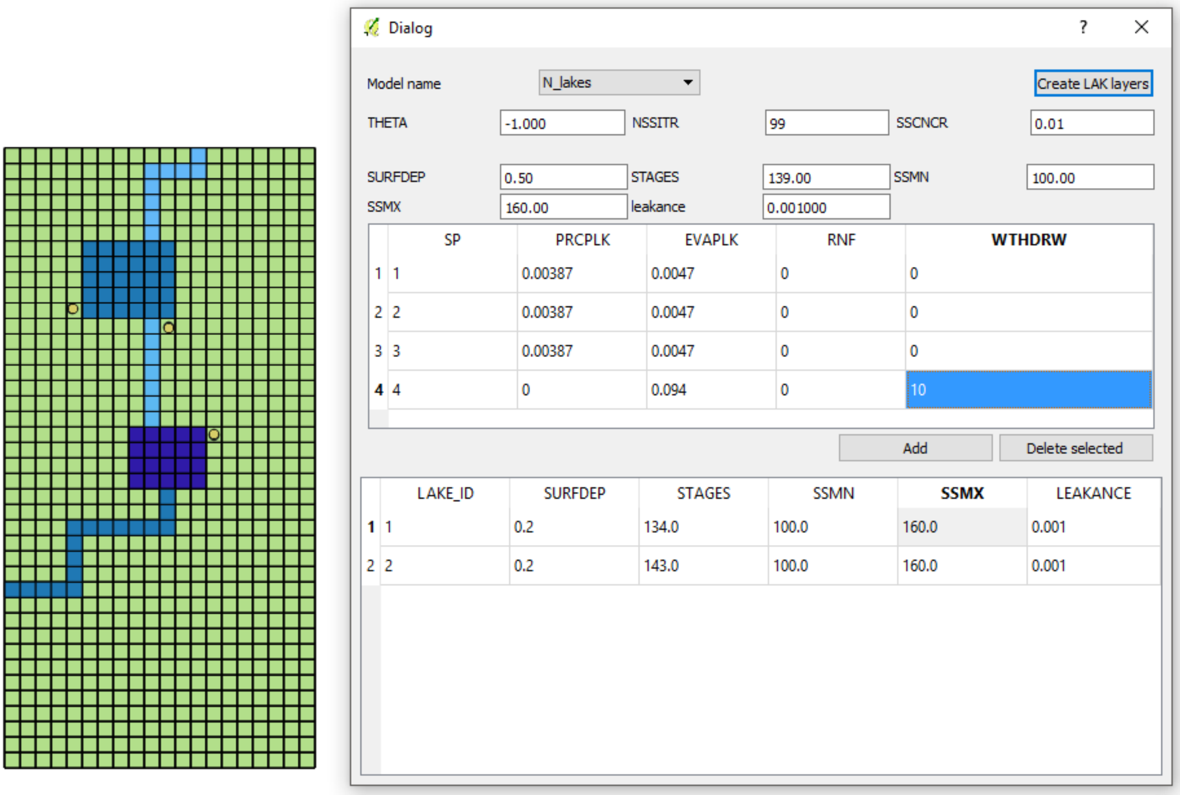
*Figure 2.1 - A simplified representation of two methods for specifying lake bathymetry.*

Supplementary sources and sinks of the lake water budget can take the form of atmospheric recharge [$LT^{-1}$], evaporation [$LT^{-1}$], overland runoff following precipitation events [$L^3T^{-1}$] and direct withdrawal [$L^3T^{-1}$]. The lake package supports several external packages, such as surface water routing with the SFR package (Niswonger & Prudic, 2010), or as surplus from the unsaturated zone flow package (Niswonger, Prudic & Regan, 2006), although the connections of lakes to these packages is always implemented within those packages rather than within the lake package, which acts as a passive reservoir.

The lake budget equation can be solved explicitly, semi- implicitly or implicitly by specifying the time-weighting factor (0, 0.5 or 1 respectively) for transient stress periods. Additionally, the number of iterations for solving the equilibrium lake stages using Newton's method can be specified, as well as the convergence criterion. Small topological variations in the lake bottom can be added to give a smooth transition for the lake equilibrium stage solution as the stage reaches the bottom of a cell; just before it dries up, or when it re-wets. The lake stage must be specified only for the initial conditions while following transient and steady state stress periods, the lake stage is determined by the lake solver. For any steady state stress periods, a minimum and maximum stage must be specified. If the lake solver determines a stage higher or lower than these limits, the simulation returns an error.

### 2.2.2 Implementation of the Lake Package in FREEWAT

In FREEWAT, the implementation of the Lake package can be generally separated into two steps. During the first step, the model geometry and active cells can be modified before the lake package itself is activated in the second. All model layer cells that contain lakes need to be specified as inactive. To conform with lake bathymetry, the vertical size of the lake cells, i.e. the thickness of the aquifer layers, can also be changed. The selection and editing of appropriate cells is performed with existing QGIS tools. Additionally, any layers which contain lakes must be convertible or unconfined and wet-able, and specified as such in the layer property flow (LPF) table, used to define the basic properties of the MODFLOW model.



Model name: N_lakes

THETA: -1.000
NSSITR: 99
SSCNCR: 0.01
SURFDEP: 0.50
STAGES: 139.00
SSMN: 100.00
SSMX: 160.00
leakance: 0.001000

| | SP | PRCPLK | EVAPLK | RNF | WTHDRW |
|---|---|---|---|---|---|
| 1 | 1 | 0.00387 | 0.0047 | 0 | 0 |
| 2 | 2 | 0.00387 | 0.0047 | 0 | 0 |
| 3 | 3 | 0.00387 | 0.0047 | 0 | 0 |
| 4 | 4 | 0 | 0.094 | 0 | 10 |

| | LAKE_ID | SURFDEP | STAGES | SSMN | SSMX | LEAKANCE |
|---|---|---|---|---|---|---|
| 1 | 1 | 0.2 | 134.0 | 100.0 | 160.0 | 0.001 |
| 2 | 2 | 0.2 | 143.0 | 100.0 | 160.0 | 0.001 |

*Figure 2.2 - The FREEWAT interface to create lake layers with a synthetic model.*

The second step requires the activation of FREEWAT's "Create lake layer" interface (Fig. 2.2) to set lake solver properties, lake source and sink terms for each time step, as well as starting stages, minimum and maximum stages, as well as lake leakance terms. Through this interface, the lake layers are created. Time constant lake information (starting stage, min and max stages, leakance and lake-bottom undulations) is stored in one model table.  Another table stores all time-variant information (precipitation, evaporation, runoff and withdrawal) for each time step and for each lake. The lake cells also need to be assigned to a specific lake using the lake ID which also specifies their lateral extent.

As a model result, the Lake package delivers the hydrologic budget summaries for the lakes for each time step specified in the MODFLOW output control. This budget for each lake includes stage, volume, volume change, precipitation, evaporation, runoff, inflow and outflow from groundwater and surface water, water use, influx from connected lakes, surface area, stage-change and percentage error, for each time step.

# 3 Solute transport

In FREEWAT the hydrological model can be coupled to a solute transport model, to simulate advective and dispersive transport of several species, both in unsaturated and saturated zone. The possibility to simulate viscosity and density dependent flow is present as well. Such capabilities are particularly relevant to approach studies on seawater intrusion processes (where density variations of water due to salinity effect are crucial), or for assessing geothermal plants at low- and medium-enthalpy.

Simulation of heat transport (also coupled with additional chemical species) is possible just treating heat as a *species* and defining diffusive coefficient and other parameters in a coherent way.

## 3.1 Conceptualization of solute transport in FREEWAT

Solute transport is solved in FREEWAT by applying the well-known MT3DMS code (A Modular Three-Dimensional Multi-species Transport Model for Simulation of Advection, Dispersion, and Chemical Reactions of Contaminants in Groundwater Systems). The User can find details on the code and documentation material (Zheng and Wang, 1999).

Furthermore, viscosity- and density-dependendent flows are solved by applying SEAWAT (Simulation of Three-Dimensional Variable-Density Ground-Water Flow and Transport), which is a revised version of coupling MODFLOW with MT3DMS, taking into account variations in density and viscosity during the numerical resolution of the groundwater flow equation (Langevin et al., 2007).

Finally, problems involving solute transport through vadoze zone are addressable in two ways:

- A simplified approach, computing an estimate of the pollutant leaching from the ground surface up to the water table: here the amount of effective concentration is then used as boundary condition for the MT3DMS code in the saturated zone. This approach is the objective of USB (Unsaturated Solute Balance) module, documented in section below.
- Activating in the transport model the package UZT (Unsaturated Zone Transport), included in the code MT3D-USGS (Bedekar et al., 2016), a USGS updated release of the groundwater solute transport code MT3DMS. Some additional remarks about this mode is reported in a specific sub-section below

The following processes active in the saturated zone (each one corresponding to a specific package of MT3DMS or SEAWAT code) are implemented in FREEWAT:

- ADV (Advection package).
- DSP (Hydrodynamic Dispersion), taking into account mechanical dispersion and molecular diffusion phenomena.
- SSM (Sinks and Sources Terms), simulating solute mass entering the model domain through sources or leaving the model domain through sinks.

- RCT (Chemical Reactions), simulating equilibrium-controlled linear or nonlinear sorption, nonequilibrium (rate-limited) sorption, and first-order reaction that can represent radioactive decay or provide an approximate representation of biodegradation. The general formulation sorption can also be used to model kinetic mass transfer between the mobile and immobile domains in a dual-domain advection-diffusion model. No reaction among different species can be considered (namely no geochemical reaction).
- VDF (Variable Density Flow)
- VCS (Viscosity-depedent Flow)

The following packages/options can not be activated directly from FREEWAT platform (to activate them, the User needs to write or modify input files generated by FREEWAT):

- RCT (Chemical Reactions) package can be activated only in case of single species simulations.
- HSS (Hydrocarbon Spill Source) package, to provide a seamless linkage to the MT3DMS transport simulator for the Hydrocarbon Spill Screening Model (HSSM) version, is not implemented in FREEWAT.
- Application of SEAWAT (density- and viscosity-dependent flow) is possible only in the saturated zone.

## 3.2 Option to activate UZT (Unsaturated Zone Transport)

UZT (Unsaturated Zone Transport) implies the application of the code MT3D-USG (Bedekar et al., 2016), coupled with the activation of UZF (Unsaturated Zone Flow) package in the flow model. The latter has to be solved by a special version of MODFLOW, namely MODFLOW-NWT, instead of the standard MODFLOW-2005. Therefore, for a successful activation of UZT package, the User is required to download this code and to input the path of its executable file within Program Locations table. The peculiarity of this procedure lies on saving flow fluxes also related to the unsaturated part, namely the one computed by UZF package (1D simulation of vadose zone, using the kinematic-wave approximation), and mapping it (along with storage changes) onto the MODFLOW grid for use by MT3D-USGS. UZT requires to input: initial saturated thickness, initial water contents, concentration associated with infiltration flux, concentration associated with evapo-transpiration flux.

The following limitations apply to UZT included in FREEWAT platform:

- UZT can be applied also to multi-species problem, but a deep testing of this applicability has not been carried out so far. Therefore, Users might experience some drawbacks on multi-species problem with UZT active: in such cases, Users are kindly requested to feedback these problems to FREEWAT Developers.
- Some testing models revealed problems in running MT3D-USGS when a significant amount of surface leakage is generated by UZF package. FREEWAT Developers are in contact with MT3D-USGS developers, who announced a new version coming soon, in which such potential problems will be solved completely. FREEWAT Users should stay tuned at https://water.usgs.gov/ogw/mt3d-usgs to follow code updates.

## 3.3 Conceptualization of USB (Unsaturated Solute Balance)

USB solves the following problem: if one (or more) area(s) on the ground surface are affected by the presence of a pollutant species, it is necessary to estimate (for each stress period in the model) how much concentration of this pollutant reaches the water table, and eventually spread out following the ground water flow. USB will compute this estimate, according to the following approach (Fig. 3.1):

1. Define the contamination zone on the top layer of the model (representing the ground surface).
2. Run the MODFLOW model including the package UZF (Niswonger et al., 2006) to estimate the top-infiltration rate, the unsaturated thickness and the aquifer recharge rate (and save such info for each cell of the model where the source is placed).
3. Take the advection equation for the unsaturated zone, and estimate the solution by means of a difference equation. This equation does not account for dispersion phenomena, while a rate of decay can be defined by the User, to consider dissolution phenomena (see details later on).
4. Take the solution (for each cell, for each stress period) as datum for the Constant Concentration term in SSM Package of MT3DMS (Zeng and Wang, 1999), the latter being the code included in FREEWAT to simulate solutes transport in groundwater, i.e. saturated part of the model domain.

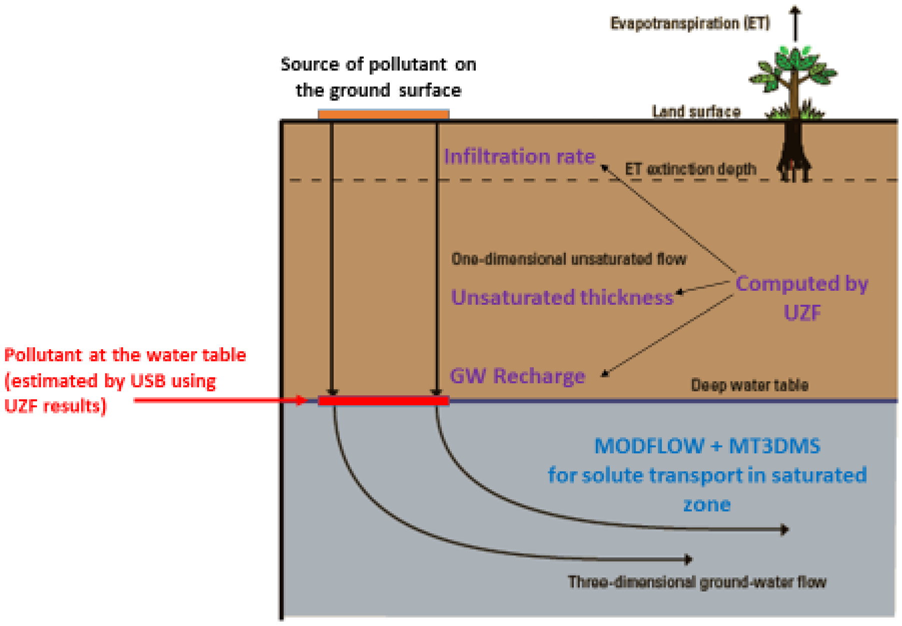

Evapotranspiration (ET)
Source of pollutant on the ground surface
Land surface
Infiltration rate
ET extinction depth
One-dimensional unsaturated flow
Unsaturated thickness
Computed by UZF
Pollutant at the water table (estimated by USB using UZF results)
GW Recharge
Deep water table
MODFLOW + MT3DMS for solute transport in saturated zone
Three-dimensional ground-water flow

*Figure 3.1 - A schematic of the coupling USB-UZF to estimate the solute concentration reaching the water table (modified from Niswonger et al., 2006).*

*Details on the process of solute transport in the vadose zone*

USB module estimates the concentration of the solute reaching the water table, by means of the following procedure.

Let us consider the pure advection equation in vadose zone (dispersion phenomena are neglected):

$$\frac{\partial(\theta c)}{\partial t} + \frac{\partial(qc)}{\partial z} = -\mu\theta c \qquad (1)$$

where:

μ is a decay coefficient [M/(L$^3$ T)],

ϑ is the water content [dimensionless],

*q* is the Darcian flux [L/T],

*c* is the concentration [M/L$^3$] (unknown).

Such an equation has to be coupled with the Richards' equation describing the water flow in the unsaturated zone (Niswonger et al., 2006). Instead of solving this coupling by means of a numerical scheme, as done by the upcoming code MT3DMS-UZF (Morway et al., 2013), we can obtain a mass balance within the unsaturated thickness: during the stress period *(t, t + dt)*, being *dt* the stress period length, eq. (1) can be read as a difference equation, namely substituting derivatives with incremental differences, with (t, t+dt) as time range and (0, z(t+dt)) as length range, respectively.

With this approach, eq. (1) leads to:

$$c_{out}(t + dt) = (1 - \mu)c_{out}(t) + \frac{dt}{z(t+dt)}[q_{in}(t + dt)c_{in}(t + dt) - q_{out}(t + dt)c_{out}(t + dt)] \ ,$$

and,

$$c_{out}(t + dt) = \frac{(1-\mu dt)c_{out}(t)+\frac{dt}{z(t+dt)\theta_S}q_{in}(t+dt)c_{in}(t+dt)}{1+\frac{dt}{z(t+dt)\theta_S}q_{out}(t+dt)} \ , \qquad (2)$$

with $c_{out}(0) = 0$ and where:

$c_{in}(t)$, initial concentration at the ground surface at time *t* (input),

$q_{in}(t)$, applied infiltration flux, at time *t* (retrieved from UZF output file),

*z(t)*, unsaturated thickness, at time *t* (retrieved from UZF output file),

$q_{out}(t)$, flux of water (recharge rate) at the water table, at time *t* (retrieved from UZF output file).

Eq. (2) applies to each cell of the top layer in which the contaminant source is present. In case of more than one components (chemical species), this procedure is replicated for each component.

*Code verification*

Formula (2) can be read as an intermediate procedure between the more accurate numerical solution of the flow/transport problem in the unsaturated zone and the simpler calculus of a *leaching factor or dissolution factor*. The latter is usually defined as the ratio between the concentration of contaminant in water (evaluated at the water table) and the fraction of mass of contaminant per unit volume of soil (at the ground surface), namely (Park and San Juan, 2000):

$$L_f = C_s/C_{gw},$$

- $L_f$, leaching factor, dims. *Kg/l*
- $C_s$, mass fraction of concentration in soil at the ground surface, dimensionless.
- $C_{gw}$, concentration in water at the water table, dims. *mg/l*

If the amount of contaminant at the ground surface is expressed as water concentration (say $C_w$) $C_s$ can be calculated as

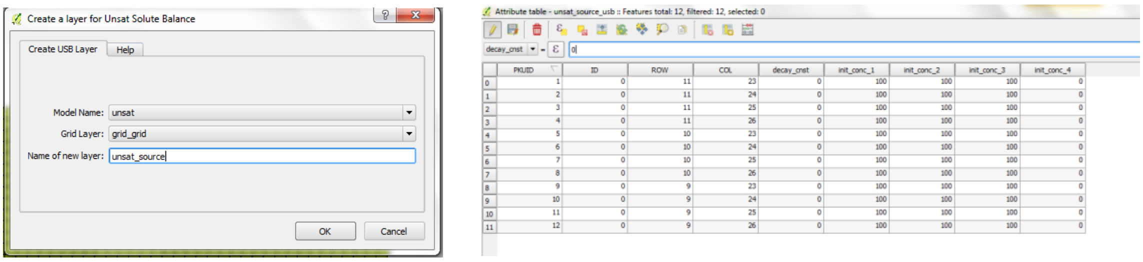
$$C_s = C_w \frac{\theta_w}{\rho_s(1-\theta_s)},$$

where $\theta_w$ , $\theta_s$ *and* $\rho_s$ are the moisture water content, the porosity and the soil bulk density, respectively.

Considering this analytical approach, we can compute the leaching factor obtained by USB. In tests made to prove the robustness of the USB method (including the exercise done during the training sessions c/o FREEWAT partners), we obtained a leaching factor ranging in the order of 0.05÷2.3 *Kg/l*, which correspond to typical values used in literature (Park and San Juan, 2000).

## 3.4 USB module in FREEWAT

The User can apply USB once a groundwater flow model has been defined, including UZF package to represent the unsaturated zone flow. The sub-menu *FREEWAT > Solute transport > Create Unsat Solute Balance Layer* opens a GUI to create the USB Model Data Object (MDO in the following), the latter been a clone of the grid, limited to the zone(s) in which a contaminant is present at the ground surface, and having as additional fields the concentration value, for each stress period (Fig. 3.2).



Create a layer for Unsat Solute Balance

Create USB Layer | Help

Model Name: unsat

Grid Layer: grid_grid

Name of new layer: unsat_source

OK | Cancel

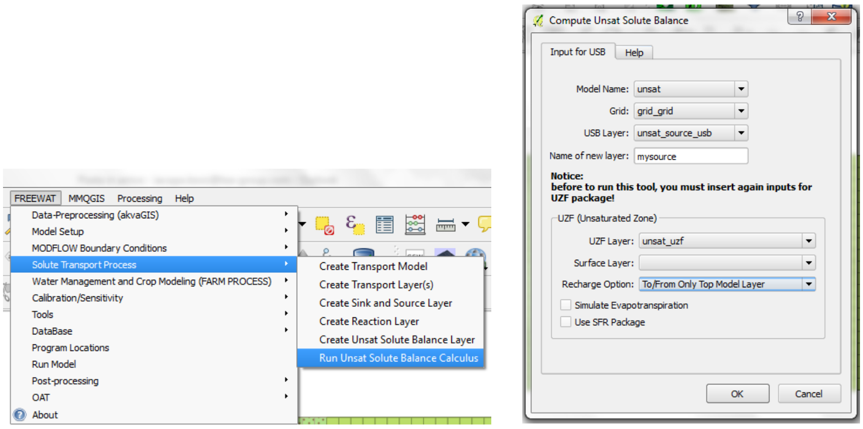
Attribute table - unsat_source_usb :: Features total: 12, filtered: 12, selected: 0

decay_cnst = 0

|  | PKUID | ID | ROW | COL | decay_cnst | init_conc_1 | init_conc_2 | init_conc_3 | init_conc_4 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 11 | 23 | 0 | 100 | 100 | 100 | 0 |
| 1 | 2 | 0 | 11 | 24 | 0 | 100 | 100 | 100 | 0 |
| 2 | 3 | 0 | 11 | 25 | 0 | 100 | 100 | 100 | 0 |
| 3 | 4 | 0 | 11 | 26 | 0 | 100 | 100 | 100 | 0 |
| 4 | 5 | 0 | 10 | 23 | 0 | 100 | 100 | 100 | 0 |
| 5 | 6 | 0 | 10 | 24 | 0 | 100 | 100 | 100 | 0 |
| 6 | 7 | 0 | 10 | 25 | 0 | 100 | 100 | 100 | 0 |
| 7 | 8 | 0 | 10 | 26 | 0 | 100 | 100 | 100 | 0 |
| 8 | 9 | 0 | 9 | 23 | 0 | 100 | 100 | 100 | 0 |
| 9 | 10 | 0 | 9 | 24 | 0 | 100 | 100 | 100 | 0 |
| 10 | 11 | 0 | 9 | 25 | 0 | 100 | 100 | 100 | 0 |
| 11 | 12 | 0 | 9 | 26 | 0 | 100 | 100 | 100 | 0 |

*Figure 3.2 - Right: GUI to create USB MDO; Left: attribute table of the MDO for USB.*

The MDO is the basis to run the USB calculus, accessible through *FREEWAT > Solute Transport Process > Run Unsat Solute Balance Calculus* (Fig. 3.3).


FREEWAT
MMQGIS
Processing
Help
Data-Preprocessing (akvaGIS)
Model Setup
MODFLOW Boundary Conditions
Solute Transport Process
Water Management and Crop Modeling (FARM PROCESS)
Calibration/Sensitivity
Tools
DataBase
Program Locations
Run Model
Post-processing
OAT
About
Create Transport Model
Create Transport Layer(s)
Create Sink and Source Layer
Create Reaction Layer
Create Unsat Solute Balance Layer
Run Unsat Solute Balance Calculus

Compute Unsat Solute Balance

Input for USB | Help

Model Name: unsat

Grid: grid_grid

USB Layer: unsat_source_usb

Name of new layer: mysource

**Notice:**
**before to run this tool, you must insert again inputs for UZF package!**

UZF (Unsaturated Zone)

UZF Layer: unsat_uzf

Surface Layer:

Recharge Option: To/From Only Top Model Layer

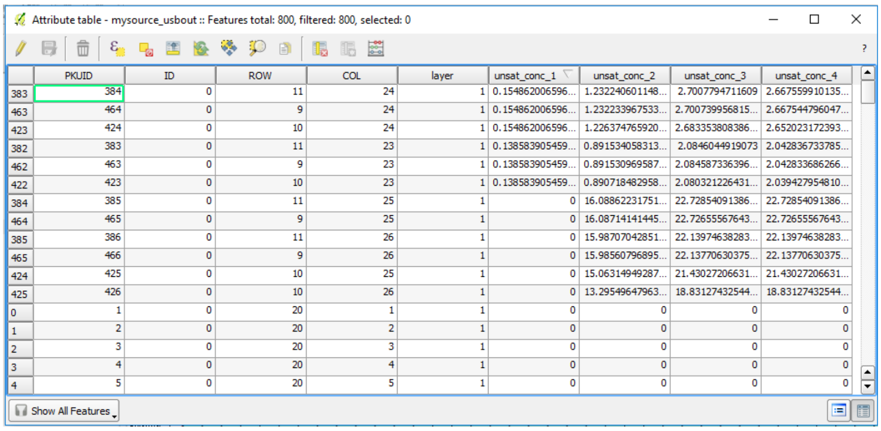
□ Simulate Evapotranspiration

□ Use SFR Package

OK | Cancel

*Figure 3.3 - Sub-menu and GUI to run the USB calculus.*

USB calculus produces another MDO (Fig. 3.4), in which the obtained value for concentration reaching the water table is saved, for each cell and for each stress period. This MDO can be used to pass such values to the MDO for SSM package of MT3DMS (sub-menu *FREEWAT > Solute Transport > Create Sink and Source Layer*), and eventually to run a transport model considering this amount of contaminant as constant source (for each cell for each stress period) active at the water table.



Attribute table - mysource_usbout :: Features total: 800, filtered: 800, selected: 0

| | PKUID | ID | ROW | COL | layer | unsat_conc_1 | unsat_conc_2 | unsat_conc_3 | unsat_conc_4 |
|---|---|---|---|---|---|---|---|---|---|
| 383 | 384 | 0 | 11 | 24 | 1 | 0.154862006596... | 1.232240601148... | 2.7007794711609 | 2.667559910135... |
| 463 | 464 | 0 | 9 | 24 | 1 | 0.154862006596... | 1.232233967533... | 2.700739956815... | 2.667544796047... |
| 423 | 424 | 0 | 10 | 24 | 1 | 0.154862006596... | 1.226374765920... | 2.683353808386... | 2.652023172393... |
| 382 | 383 | 0 | 11 | 23 | 1 | 0.138583905459... | 0.891534058313... | 2.0846044919073 | 2.042836733785... |
| 462 | 463 | 0 | 9 | 23 | 1 | 0.138583905459... | 0.891530969587... | 2.084587336396... | 2.042833686266... |
| 422 | 423 | 0 | 10 | 23 | 1 | 0.138583905459... | 0.890718482958... | 2.080321226431... | 2.039427954810... |
| 384 | 385 | 0 | 11 | 25 | 1 | 0 | 16.08862231751... | 22.72854091386... | 22.72854091386... |
| 464 | 465 | 0 | 9 | 25 | 1 | 0 | 16.08714141445... | 22.72655567643... | 22.72655567643... |
| 385 | 386 | 0 | 11 | 26 | 1 | 0 | 15.98707042851... | 22.13974638283... | 22.13974638283... |
| 465 | 466 | 0 | 9 | 26 | 1 | 0 | 15.98560796895... | 22.13770630375... | 22.13770630375... |
| 424 | 425 | 0 | 10 | 25 | 1 | 0 | 15.06314949287... | 21.43027206631... | 21.43027206631... |
| 425 | 426 | 0 | 10 | 26 | 1 | 0 | 13.29549647963... | 18.83127432544... | 18.83127432544... |
| 0 | 1 | 0 | 20 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 2 | 0 | 20 | 2 | 1 | 0 | 0 | 0 | 0 |
| 2 | 3 | 0 | 20 | 3 | 1 | 0 | 0 | 0 | 0 |
| 3 | 4 | 0 | 20 | 4 | 1 | 0 | 0 | 0 | 0 |
| 4 | 5 | 0 | 20 | 5 | 1 | 0 | 0 | 0 | 0 |

*Figure 3.4 - Attribute table of the MDO containing results of USB calculus.*

# 4 Water Management Module

FREEWAT includes capabilities to address water resource management. Their specific goal is to exploit the output of a numerical simulation to solve problems of water delivery control and optimization. This can be successfully obtained by upscaling the grid cell-based simulation results, to get budgets of water demand and usage, focused on one (or more) zone(s) of the model domain.

Conjunctive use of water is the joint usage and management of surface- and groundwater resources to meet required water demand and minimize potential damage to the quantity or quality of the resource. Therefore, to get an effective representation of conjunctive use of surface and subsurface water, it is necessary to integrate simulation methods for subsurface, surface, and urban and agricultural water-demand computations. Furthermore, these models have to take into account the cases where there is not enough water supply to meet the total water demand, and propose possible management strategies to cope with this problem.

FREEWAT simulates water management issues by applying MODFLOW-OWHM (Hanson et al., 2014), which can be used also to include the specific computation of water demand coming from rural environments and crops acreage.

## 4.1 Conceptualization of Water Management in FREEWAT

### 4.1.1 Water Demand Units

The basic concept in MODFLOW-OWHM is the Water Demand Unit (WDU), namely any “entity” consuming water (urban zone, industrial zone, farms, rural areas, natural vegetation areas, etc.). At each WDU the User can associate one or more source/sink of water (i.e. demand or supply terms), varying in time. At the end of the simulation, a specific budget for each WDU is produced in addition to the global budget for the entire model. Therefore, the User can analyze a WDU budget as well as a comparison among different WDUs. From a modeling point of view, a WDU is a model sub-region, made by a cluster of cells of the first model layer (top layer). Thus, the whole grid is divided in one or more WDU by assign to each cell a WDU ID (namely, each cell is associated to an only and only one WDU ID). In addition, the part of the grid not really occupied by any WDU should be identified with a WDU ID.

### 4.1.2 Terms of Water Demand

For each WDU, a total water demand is input or computed, with the possibility to account of several ways for defining different terms of water demand, namely:

- Crops water demand (in case of agricultural areas or natural vegetation areas are modeled). This demand is computed as ET representing the target crop consumptive use to meet. Defining such water demand term is essential whenever an agricultural/irrigation problem is addressed. However, the entity “crop” can represent not only real crops, but also any land-use type (urban environment, water bodies, etc.) which are defined as “virtual crops” (Schmid et al., 2006; Schmid et al., 2009; Hanson et al., 2014).

- Municipal and industrial urban water demand is user-specified as “negative supplies” (specified as non-routed deliveries, see later on).
- The Total Delivery Requirement is defined as the portion of crop demand that is not met by precipitation and uptake from groundwater, increased by the inefficiency losses from irrigation.

### 4.1.3 Terms of Water Supply

For each WDU, for each stress period, the code attempts to satisfy the Total Demand Requirement with one or more delivery terms, according to the following ranking:

- 1st priority by the non-routed supply components (water transfer to/from a farm from several kinds of sources, without simulating the process of conveyance).
- 2nd priority by the semi- and fully routed deliveries (surface water: water transfer to/from a WDU through a streamflow-routing network (fully routed) or specifying diversion points from the main channel (semi-routed). The streamflow-routing network is simulated through the SFR package in MODFLOW.
- 3rd priority: ground-water pumping (wells) necessary to satisfy the total delivery requirement.

### 4.1.4 Constraints on water supply

The User has the possibility to specify, for each WDU, surface- or groundwater allotments, to represent water-rights.

Allocation constraints representing water-rights hierarchies can be imposed as surface-water allotments. These surface- water allotments may be introduced into a model a as water rights calls per stress period in units of [L3/T] for a prior appropriation system of ranked individual WDUs.

Furthermore, the simulation of a groundwater allotment for each WDU is also possible. For each WDU is given a volumetric-rate constraint that is the portion of the unit that can be derived from groundwater sources. Each allotment can represent any kind of physical or governance limit, such as a groundwater right or a transboundary operating agreement. Such limits on groundwater abstraction are useful for the assessment of limits of groundwater supply such as those imposed by development of a basin management plan or sustainability analysis that are subject to other limits to groundwater supply from secondary effects (e.g. land subsidence, seawater intrusion, streamflow capture, maintenance of groundwater-dependent ecosystems, or adaptation schemes for climate change and climate variability).

Within a WDU, groundwater allotments refer to the entire WDU. In the meantime, for each well associated to a WDU, the User imposes a maximum for the pumping rate. This is an additional option to impose specific water rights to a single WDU, but associated to a specific location.

### 4.1.5 Optimization algorithms in case of Deficit Scenario

Once the water demand and supply have been computed (for each stress period), taking into account possible constraints on surface- or groundwater delivery, the code compare the two terms Demand and Supply.

If Demand is greater than Supply, than that WDU is in a deficit scenario. In this case, the code allows estimating optimal distributions of supply components to cope with this deficit.

The User can apply two different optimization methods:

1. Crop priority-based (*water-stacking*): the User defines a priority ranking for the crops (priority crops are crops which generally are not fallowed).
    - ➢ The water supply is distributed according to the priority ranking
2. Economic optimization: User defines costs of water supply sources and profits related to crops production.
    - ➢ Then, the water supply is distributed according to the ranking of profitability. Such a ranking is established by solving an optimization problem (linear programming technique), in which the objective function is the sum of costs-benefits balance, for each crop and on each cell.

In Fig. 4.1, a workflow is reported, describing how the code proceed in case of deficit scenario, when one of the two optimization options is activated. The User is also free to select no optimization procedure: in this case, the supply is just limited to the real available water amount.

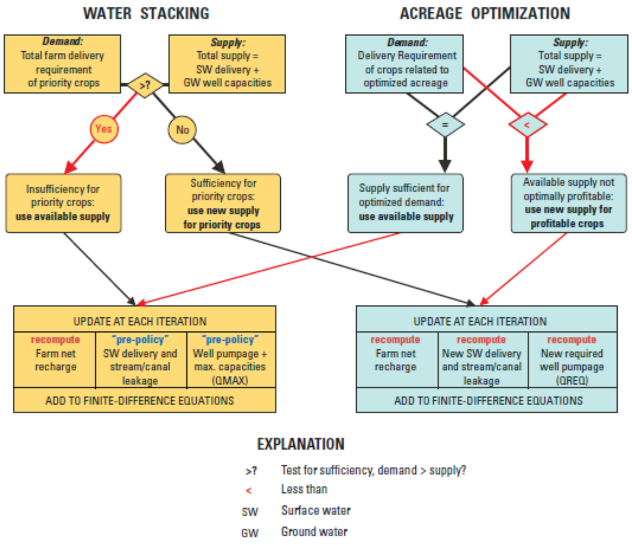

WATER STACKING
Demand: Total farm delivery requirement of priority crops
>?
Supply: Total supply = SW delivery + GW well capacities
Yes
No
Insufficiency for priority crops: use available supply
Sufficiency for priority crops: use new supply for priority crops
UPDATE AT EACH ITERATION
recompute Farm net recharge
"pre-policy" SW delivery and stream/canal leakage
"pre-policy" Well pumpage + max. capacities (QMAX)
ADD TO FINITE-DIFFERENCE EQUATIONS
ACREAGE OPTIMIZATION
Demand: Delivery Requirement of crops related to optimized acreage
Supply: Total supply = SW delivery + GW well capacities
=
<
Supply sufficient for optimized demand: use available supply
Available supply not optimally profitable: use new supply for profitable crops
UPDATE AT EACH ITERATION
recompute Farm net recharge
recompute New SW delivery and stream/canal leakage
recompute New required well pumpage (QREQ)
ADD TO FINITE-DIFFERENCE EQUATIONS
EXPLANATION
>? Test for sufficiency, demand > supply?
< Less than
SW Surface water
GW Ground water

*Figure 4.1 - Procedure applied to update source-term flow, depending on the deficit-response policy selected by the User. From Schmid et al., 2006.*

A summary of water demand and supply, along with specification of possible deficit scenarios, it is reported in Fig. 4.2.

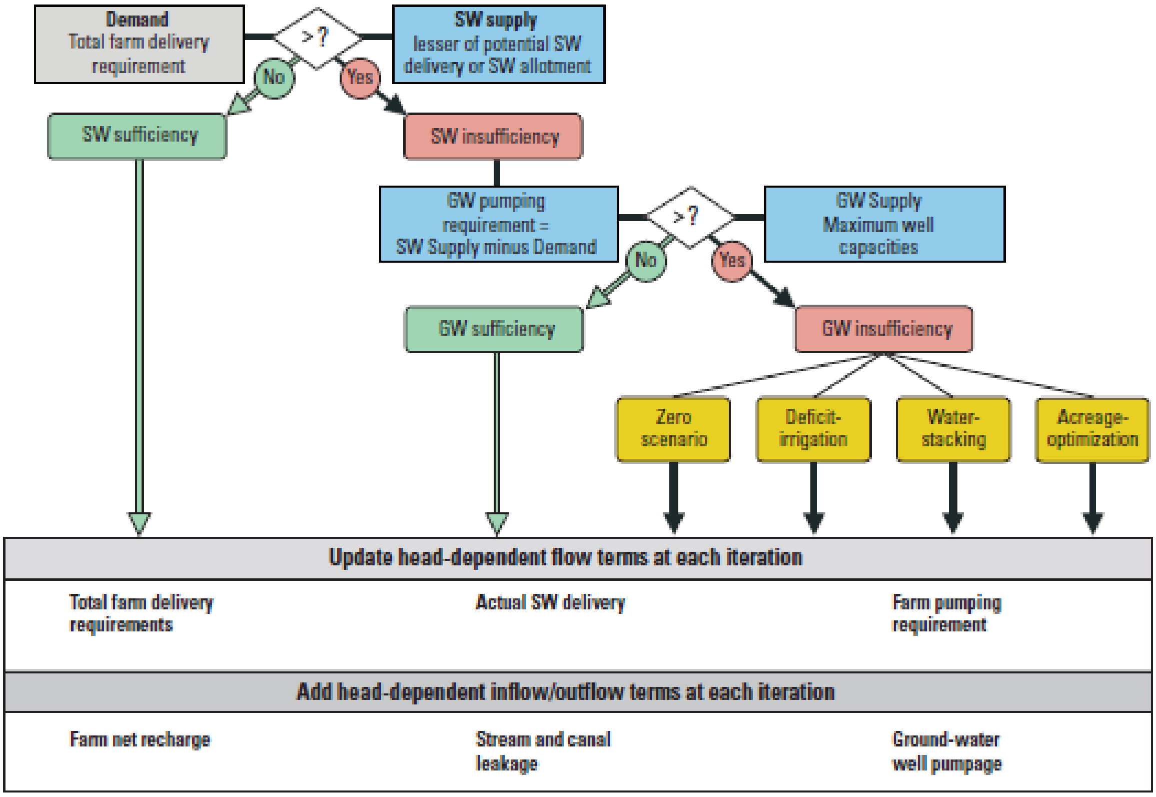
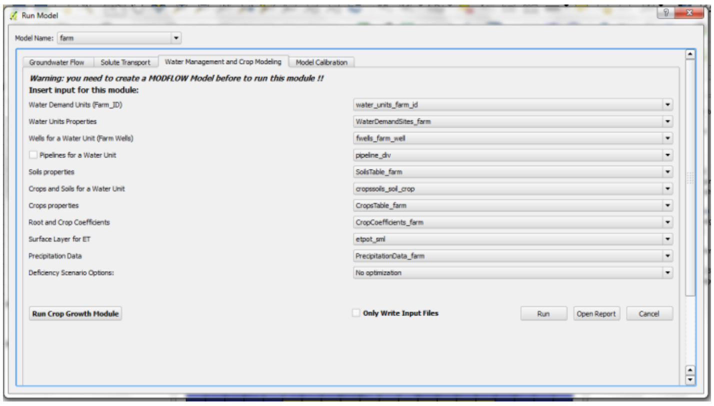

Demand
Total farm delivery requirement
>?
SW supply
lesser of potential SW delivery or SW allotment
No
Yes
SW sufficiency
SW insufficiency
GW pumping requirement = SW Supply minus Demand
>?
GW Supply
Maximum well capacities
No
Yes
GW sufficiency
GW insufficiency
Zero scenario
Deficit-irrigation
Water-stacking
Acreage-optimization
Update head-dependent flow terms at each iteration
Total farm delivery requirements
Actual SW delivery
Farm pumping requirement
Add head-dependent inflow/outflow terms at each iteration
Farm net recharge
Stream and canal leakage
Ground-water well pumpage

*Figure 4.2 - A summary of water demand, supply and method to cope with deficit scenarios. From Schmid et al., 2006.*

## 4.2 Implementation in FREEWAT

Water management capabilities of FREEWAT can be applied once the hydrological model (say the standard MODFLOW model) has been set up and run. At this stage, the User should be confident on the key modeling features, namely: model convergence, parameters values (potentially determined/analyzed by means of the sensitivity and calibration tools), consistency of the model water budget. Assuming that the model has been finalized correctly, the User can proceed to input the information needed to run management tools in MODFLOW-OWHM. All these functionalities are accessible through the sub-menu *FREEWAT > Water Management and Crop Modeling (FARM PROCESS).* More in detail, the following steps are needed:

1. Defining the classification of the model domain in WDUs.
2. Setting the properties for each WDUs (efficiency, water delivery costs, water allotments, etc.)
3. Linking the model wells (if any) to WDUs, including for each of them a maximum value for the pumping rate: this procedure defines the groundwater pumping availability and constraints, for selected WDUs.
4. Linking pipeline diversions to WDUs (if any): this procedure defines the semi-routed water delivery associated with selected WDUs.
5. Defining soils and crops distribution on the top model grid.
6. Defining soils properties, for each soil defined in the previous step: capillary fringe value.
7. Defining crops properties, for each crop defined in the previous step, such as irrigation and ET losses efficiency coefficients, profits from crop production, crop coefficient, root depth, etc.
8. Setting precipitation and reference evapotranspiration data.
9. Selecting model options and run the model (Fig. 4.3).
10. Analyzing the WDUs budget by plotting results printed in MODFLOW-OWHM output files (*FREEWAT > Post-processing > Plot budgets for Water Unit*).



Model Name: farm

*Warning: you need to create a MODFLOW Model before to run this module !!*

**Insert input for this module:**

Water Demand Units (Farm_ID): water_units_farm_id

Water Units Properties: WaterDemandSites_farm

Wells for a Water Unit (Farm Wells): fwells_farm_well

□ Pipelines for a Water Unit: pipeline_div

Soils properties: SoilsTable_farm

Crops and Soils for a Water Unit: cropssoils_soil_crop

Crops properties: CropsTable_farm

Root and Crop Coefficients: CropCoefficients_farm

Surface Layer for ET: etpot_sml

Precipitation Data: PrecipitationData_farm

Deficiency Scenario Options: No optimization

□ Only Write Input Files

*Figure 4.3 - Screen shot of the GUI to run the Water Management capabilities in FREEWAT.*

# 5 AkvaGIS (Hydrochemical Analysis Tools and Hydrogeological Analysis Tools)

This section describes specific tool included in FREEWAT to optimize the process of build the conceptual model using hydrochemical data (***Hydrochemical Analysis Tools, sub-module of AkvaGIS tools***).

The quality of the groundwater may be adversely affected by a large number of factors, such as industrializations, urbanization, irrigation, etc. (Foster, 2001; Vázquez-Suñé et al., 2005). Thus, a comprehensive evaluation of the negative impacts of these potentially hazardous activities is key on the protection of groundwater bodies and ecosystems associated.

To ensure compliance with standard regulatory guidelines (with a special focus on the requirement deriving from the GWD), continuous monitoring, evaluation, and interpretation of a large number of physical and chemical parameters are required. Such data sets need to be assessed and interpreted by water agencies, stakeholders and assessors to provide answers to questions such as: (a) the processes controlling the chemical composition of groundwater and the corresponding spatial and temporal distribution, (b) evaluation of the current groundwater quality and the achievement of good chemical status based on national thresholds of the water quality (e.g. Water Frame Directive (WFD), 2009) or (c) the regional background composition of groundwater (Mendizabal and Stuyfzand, 2009), among others.

New capabilities have to develop to face: (i) the need to manipulate large data sets collected over many years (Velasco et al., 2014); (ii) the integration of data stemming from diverse sources and gathered with different data access techniques and formats; (iii) the management of data with varying temporal and spatial extent; and (v) the integration of groundwater quality information with other relevant information such as further hydrogeological data (e.g. heads) and the pre-processing of these data in particular for the realization of groundwater models.

To face these difficulties, the *Hydrochemical Analysis Tools* sub-module was created to complement the functionalities of the FREEWAT platform in the QGIS environment (http://qgis.org). The selected reference version of QGIS is the latest LTR (Long Term Release), namely QGIS 2.14.

It is worth mentioning that any test performed so far with subsequent versions (e.g. 2.16 and 2.18) worked without experiencing any problem.

The ***Hydrochemical Analysis Tools*** sub-module is part of the AkvaGIS module, perfectly integrated into the FREEWAT platform. It is composed by a geospatial database implemented in Spatialite and a set of tools for improving the harmonization, integration, standardization, visualization and interpretation of hydrochemical data. These tools include different instruments that cover a wide range of methodologies for querying, interpreting, and comparing groundwater quality data and facilitates the pre-processing analysis for being used in the realization of groundwater modelling. Some of the tools developed are: ionic balance calculations, chemical time-series analysis, correlation of chemical parameters, and calculation of various common hydrochemical diagrams (Salinity, Schöeller-Berkaloff, Piper, and Stiff), among others. Furthermore, the sub-module allows the generation of maps of the spatial distributions of parameters, diagrams and thematic maps for the parameters measured in the queried area and classified according to the threshold approach established by a given guideline, e.g. WFD.

This set of tools are part of a wider framework developed into FREEWAT (the module pre-processing tools; AkvaGIS) created to facilitate the pre-processing of further hydrogeological analysis and interpretations. The entry point to start using both set of tools is the *Spatialite database* which contains the hydrochemical and the hydrogeological spatio-temporal information to be represented or analyzed.

Further details on the platform usage are reported in the User Manual which has been identified as User Manual (Volume 4). Details on the Python code structure and third party libraries used and dependencies are given in Appendix F.

## 5.1 Conceptualization of (AkvaGIS sub-module)

This sub-module of AkvaGIS, module of FREEWAT, was designed bearing in mind the different tools and methodologies that the water managers use to perform a comprehensive hydrogeochemical analysis and quality issues including data check, diagrams and ionic ratios, visualization, pre-processing and interpretation of the hydrochemical data. We first present the technical requirements and specifications, and in the following section (section 3) how these specifications were implemented in the final module of FREEWAT.

An effective methodology to assess hydrogeochemical data in groundwater modelling and groundwater quality controls should meet the following technical requirements:

1. Storage, integration and management of spatial features and time dependent data on a geospatial database, supporting:

1.1. Management of different data derived from both analyses of water samples at the laboratory and field data.

1.2. Integration of different types of information (e.g. normative thresholds established for a given hydrochemical parameter, heads, aquifer geometries, etc.).

1.3. Standardization and harmonization of data, including specific mechanisms to facilitate data transcription, management of different formats, edition of data and unit conversion.

1.4. Exportation of stored hydrochemical data in different formats such as .ods format, to be used for further reporting or calculations in external software (e.g. openOffice).

2. Data processing and analysis of the data stored in the geospatial database using:

2.1. GIS environment (e.g. QGIS) which provides tools to: (1) estimate/validate the spatial distribution of the chemical/physical components; (2) integrate different types of data settings; (3) create interactive mapping; (4) perform an effective assessment of consistency and correlation of the input data; (5) allow the spatial analysis with external analytical tools.

2.2. Specific tools to facilitate the hydrogeochemical analysis by using data quality control and conventional graphical analysis techniques (e.g. Stiff diagram).

2.3. Tools to facilitate the generation of maps showing the spatial distributions of different physic-chemical parameters and thematic maps for the parameters measured in the queried area and classified according to the threshold approach established by a given guideline (e.g. Water Framework Directive.)

2.4. Tools to facilitate the spatial-temporal queries and visualization of the hydrochemical data (e.g. creation of time evolution plots).

2.5. General tools of pre-processing (e.g. detection and visualization of outliers using automatic calculation of some descriptive statistic such as maximum, minimum, mean values, etc.) and validating data (e.g. deletion of duplicates and of obvious transcription errors).

## 5.2 Hydrochemical and hydrogeological analysis tools in FREEWAT

The requirements enumerated in the previous section were adopted as guidelines during the design of the present module of FREEWAT.

First, a geospatial database, specially designed to manage and store the hydrochemical data was developed (AkvaGIS Database). A set of tools for managing, visualizing, analyzing, interpreting and pre-processing these data were created and implemented in the FREEWAT platform (Fig. 5.1).

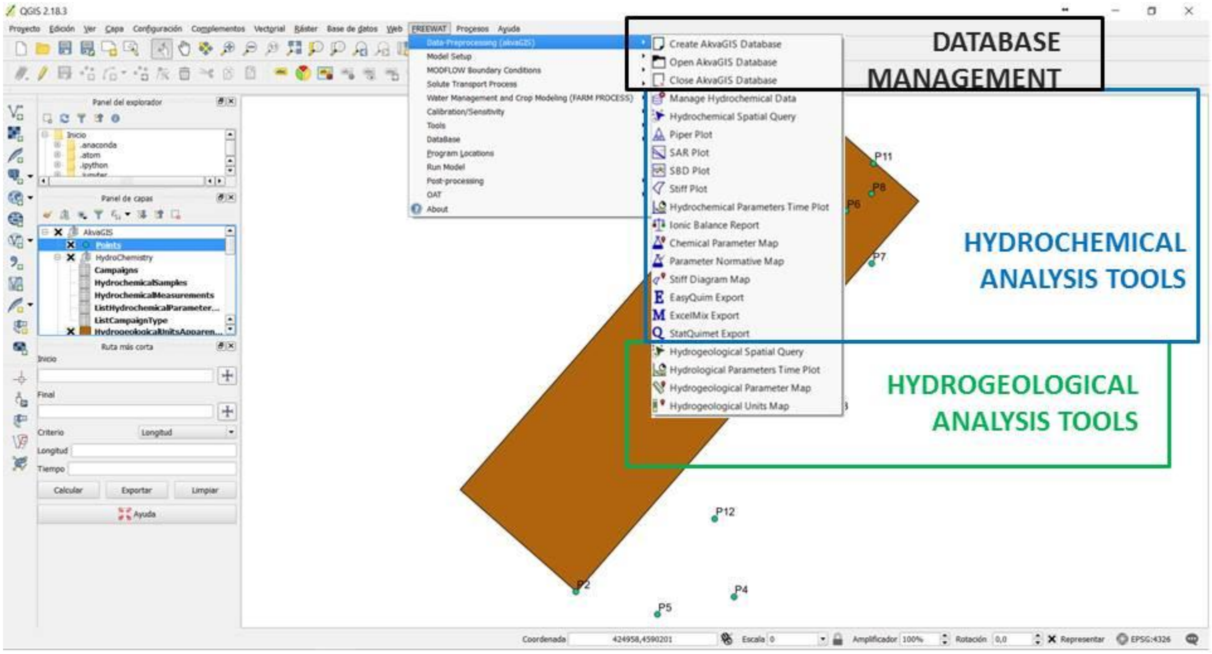


*Figure 5.1 - Sketch representing the different submodules of data pre-processing tools (AkvaGIS) implemented in the FREEWAT platform in QGIS. Notice that Hydrochemical Analysis Tools are highlighted in blue colour.*

These tools and its main capabilities are described in the following subsections.

### 5.2.1 Database management tools

The AkvaGIS database was implemented in *RDBMS Spatialite*. Its structure facilitates: 1) the data standardization and harmonization, 2) the storage and management of large amount of spatial features and time-dependent data and 3) the creation and the execution of queries. This database has been designed to include a wide range of information related to hydrochemistry and hydrogeology. Fig. 5.2 shows the database scheme in terms of the hydrochemical tables.

Regarding the hydrochemical data, the database enables the user to include organic and inorganic chemical records, as well as other relevant measured parameters (temperature, Eh, PH, etc.). Further information about normative (e.g. WFD), field campaigns, and laboratory experiments can also easily be included.

A complete description of the data content, structure and format of each element of the database is described in the User Manual (Volume 4).

In the following, the hydrochemical tools inside the AkavaGIS are briefly described:

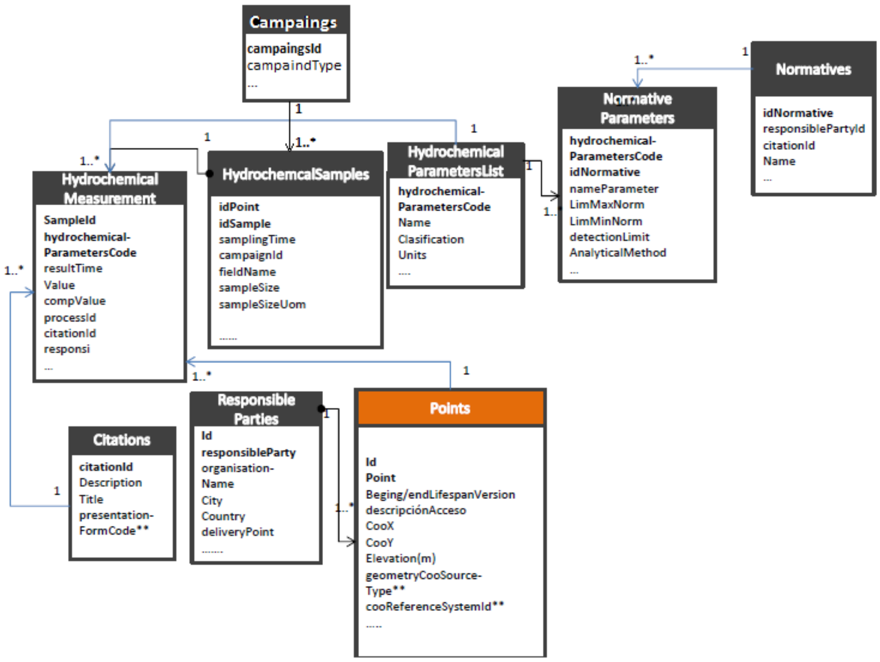


*Figure 5.2 - Simplified conceptual diagram representing the main contents of hydrochemical data in the hydrogeological database. The 1 and 1* represents the cardinality of the relationship between tables.*

As occurs with the Hydrochemical Analysis Tools, the ***Hydrogeological Analysis Tools*** query and visualize the different data stored in a geospatial database (AkvaGIS database) specially designed to manage and store the hydrochemical and the hydrogeological data. This database was implemented in *RDBMS Spatialite* and its structure facilitates: 1) data standardization and harmonization, 2) storage and management of large amounts of spatial features and time-dependent data, and 3) creation and execution of simple queries.

Regarding the hydrogeological data, the database enables the user to introduce hydrogeological information such as wells descriptions (e.g. depth, curb, diameter, screen, etc.) and hydrogeological spatio-temporal data (e.g. well abstraction or piezometric heads). Moreover, AkvaGIS database contains spatial entities and attribute tables to describe and represent different hydrogeological units. Thus, the user can improve the conceptual model and it help to perform the hydrogeological model. Fig. 5.3 shows a scheme regarding to hydrogeological tables.

In order to ensure the standardization, harmonization and interoperability of the data, several code list and schemas of some features were created taking into account standard guidelines (e.g. INSPIRE 2011, 2013, OGCWaterM.L2.0; OGC, 2012).

A complete description of the data content, structure and format of each element of the database is detailed described in the as User Manual version (Volume 4).

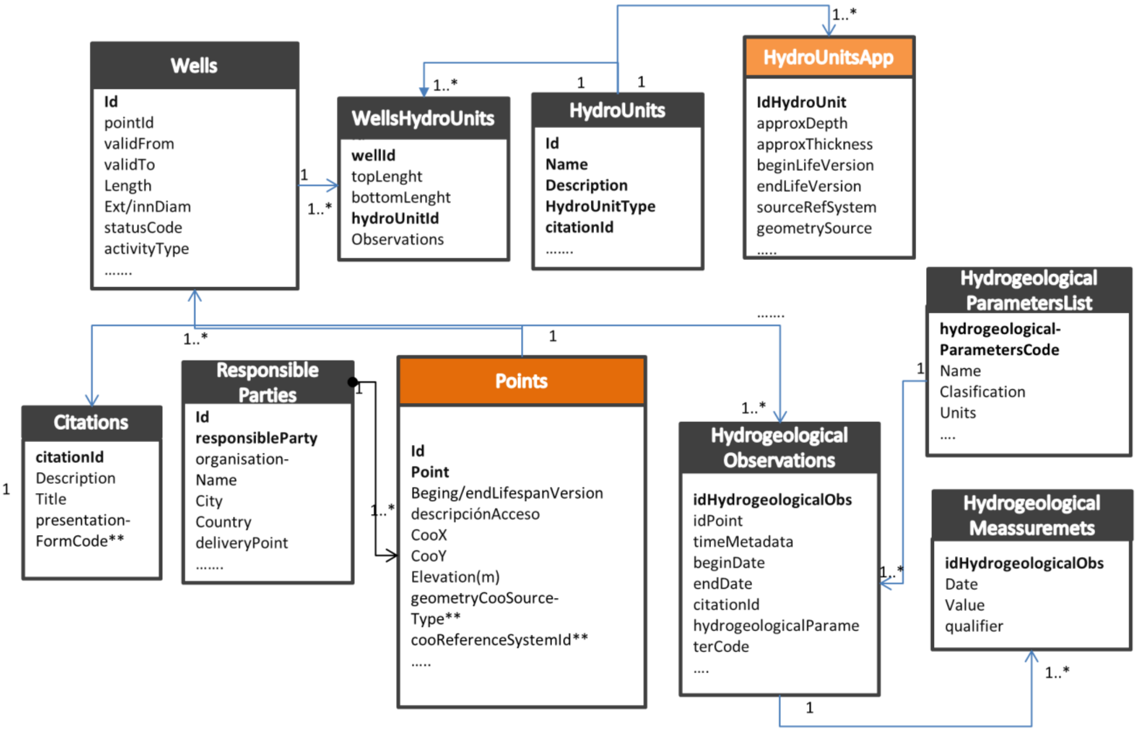


*Figure 5.3 - Simplified conceptual diagram representing the main contents of hydrogeological data in the database. The 1 and 1* represents the cardinality of the relationship between tables.*

The ***Database Tools*** included in the data-preprocessing Tools of AkvaGIS, is a set of commands that enable us to create, open or close an AkvaGIS SpatiaLite database where the hydrogeological data will be introduced to be queried by ***Hydrogeological Analysis Tools*** (see Fig. 5.1, Database Management Tools). The management and introduction of data into the database can be done using the different wizards provided by the inherent tools included in QGIS.

***Manage Hydrochemical Data***: allows to modify, insert (one by one or massively using .cvs formats) and visualize hydrochemical data from existing points stored in the database.

### 5.2.2 Hydrochemical Spatial Query

***Hydrochemical Spatial Query:*** This command aids to quickly check data related to laboratory and field campaign samples obtained from selected points in the screen for an specific time interval. The hydrochemical spatial Query is created and stored in the database, ready to apply be applied in other hydrochemical tools of the Hydrochemical Analysis Tools sub-module (e.g. Piper diagram, Stiff diagram, ionic mass balance, etc.). Note that this tool enable the visualization and management of the censored values (which are the concentrations of some elements reported as “non-detected”, “less-than” or “greater-than”) allowing conversion of censored values to numerical values that can be further included back into the database.

### 5.2.3 Ionic Balance calculations

⚖ ***Ionic Balance Report:*** This command was created to calculate ionic balance automatically for each selected sample stored in the database. Once the query is done, the user can save the results in a table or in an ionic balance report in .ods format.

### 5.2.4 Hydrochemical Diagrams

This set of tools was designed to facilitate the creation of standard hydrogeochemical diagrams for interpreting groundwater chemical analysis. Piper, Salinity diagrams, Schöeller-Berkaloff, and modified Stiff diagrams can be created automatically for the chosen dataset (only if the parameters necessary for the creation of each diagram are available). Also, the user can edit the plot setup (plot size, Point style, legend, etc.).

In order to illustrate the capabilities of this set of tools, the results of an applied case (Barcelona, Spain)[1] are shown in Fig. 5.4. The case is related to the query of a set of sampling points using the Hydrochemical diagrams command for a test period of 3 years.

In the following the tools are briefly described, together with the applied case examples.

[1] The results are presented only for illustration purposes, and they lack hydrochemical validity.

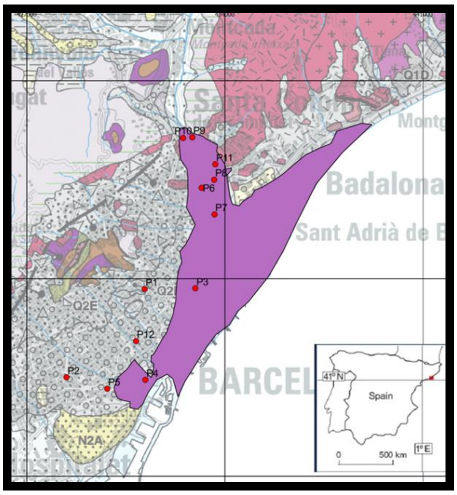


*Figure 5.4 - Map of the study area showing the sampling points chosen as representative for illustrating the capabilities of hydrochemical diagrams tools of AkvaGIS.*

***Piper Plot*:** To create Piper Plots for a selected query. Fig. 5.5 shows the resulting Piper diagram for a group of sampling points located in the study area.

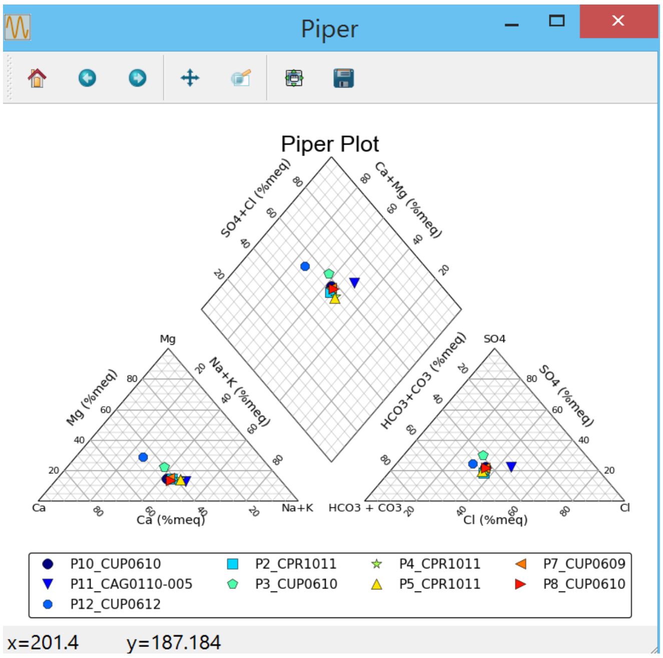


*Figure 5.5 - Piper diagram obtained with the command Piper Plot.*

***SAR Plot*:** This command creates Salinity Diagrams of the selected query. Fig. 5.6 shows the resulting SAR Plot diagram for a group of sampling points located in the study area.

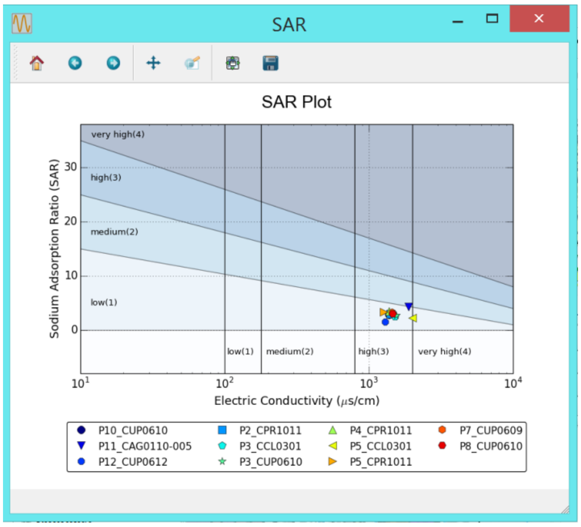


*Figure 5.6 - SAR diagram obtained with the command SAR Plot.*

***SBD Plot***: This command creates Shöeller-Berkaloff diagrams of the selected query. Fig. 5.7 shows an example of a SBD plot.

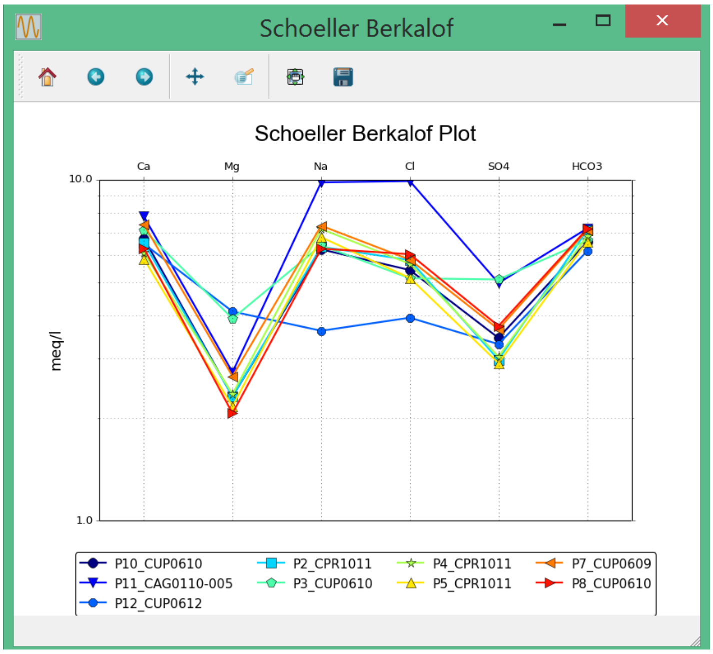


*Figure 5.7 - Shöeller-Berkaloff Diagram obtained with the command SBD Plot.*

***Stiff Plot***: This command creates Stiff plots of the selected query (Fig. 5.8). The stiff diagram results can be saved as .svg files.

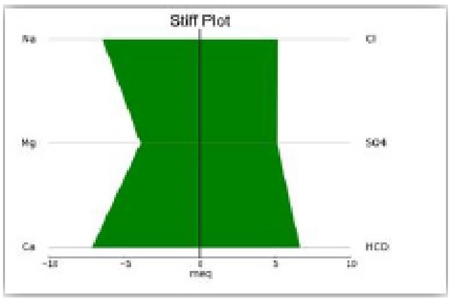


*Figure 5.8 - Stiff diagram for one sampling point of the study area.*

*Stiff Diagram map*: This command enables to obtain general statistical parameters (minimum, maximum or mean value) of the selected group of data for being mapped in the stiff map (Fig. 5.9). The results can be saved as .ods format.

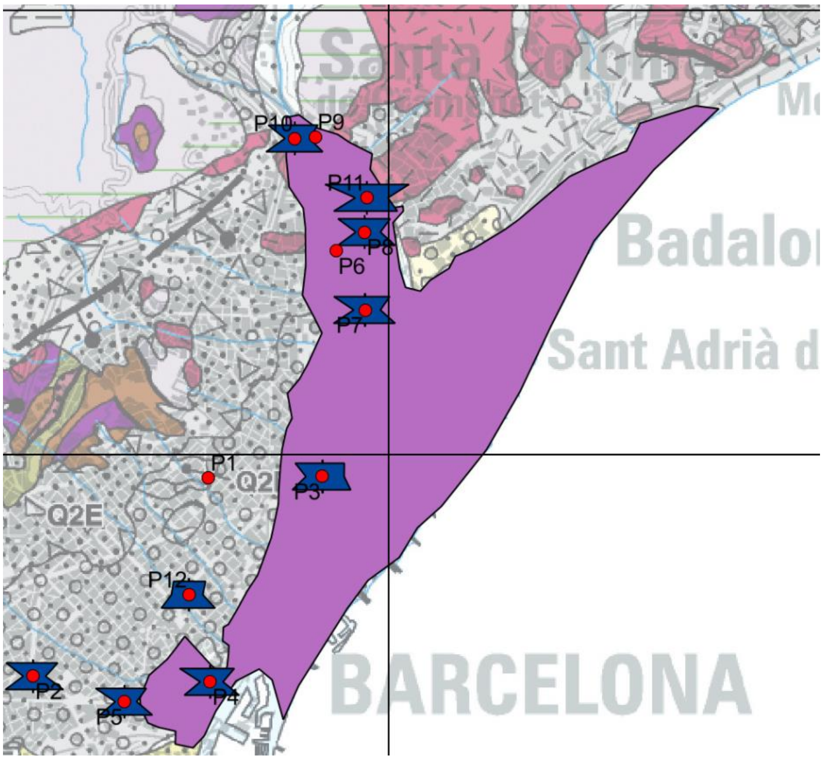


*Figure 5.9 - Stiff Maps obtained with the Stiff Diagram Plot Tool.*

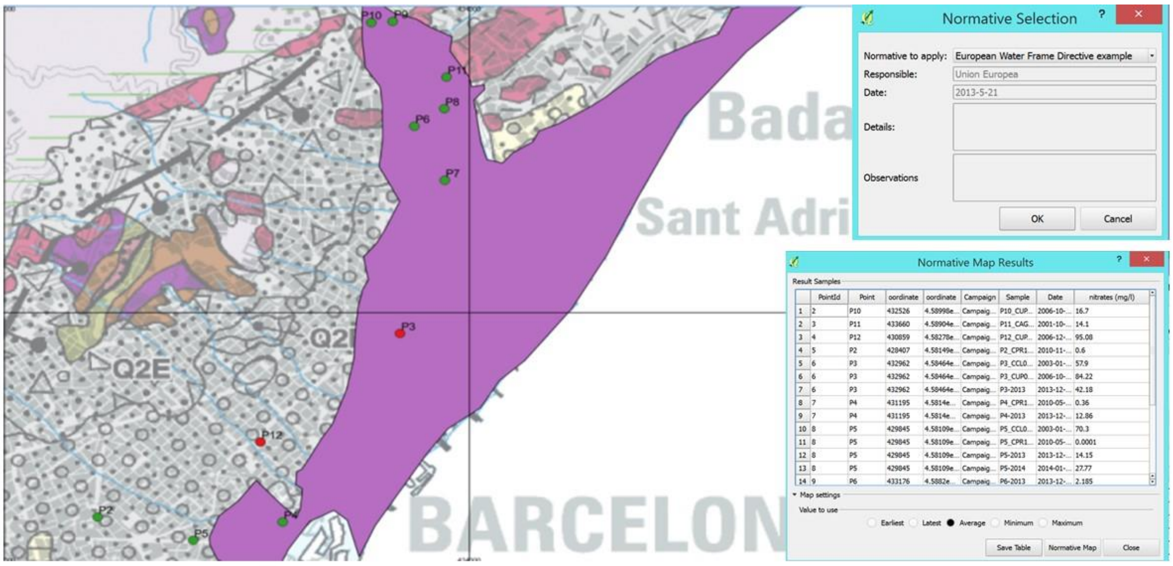
### 5.2.5 Hydrochemical Maps and Time plots

These tools enable to use the following range of methodologies and calculations for querying, interpreting and comparing groundwater quality parameters:

*Time Plot*: To create time plots of selected parameters of a selected query (previously created with the *Hydrochemical Spatial Query* tool). Henceforward, the results are saved in a table (e.g. as .csv, .ods formats) and automatically a time plot is performed.

*Chemical Parameter Plot*: Use this command to obtain thematic maps for the selected physic-chemical parameters. This tool computes general statistical parameters such as earliest, latest, average, minimum, maximum or mean values for each selected parameter for a given period of time and for a point or a group of selected points. These statistical values can be represented in a map as point entities. Results can be easily used for further spatial analysis in the same QGIS platform. In addition, the results of this query can be exported to external platforms (e.g. .ods format).

*Parameter Normative Map*: This command enables to obtain maps for the queried parameters, classified according to the threshold values established by a given guideline (e.g. Water Framework Directive). User can choose the Normative (different normative and threshold values must to be introduced previosly in the database) to obtain a thematic map with the selected parameters. The results can be saved. Fig. 5.10 shows a resulting map of Nitrates classified according to a given normative (e.g. WFD for Nitrates).



Normative Selection

Normative to apply: European Water Frame Directive example

Responsible: Union Europea

Date: 2013-5-21

Details:

Observations

Normative Map Results

Result Samples

| | PointId | Point | oordinate | oordinate | Campaign | Sample | Date | nitrates (mg/l) |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | P10 | 432526 | 4.58998e... | Campaig... | P10_CUP... | 2006-10-... | 16.7 |
| 2 | 3 | P11 | 433660 | 4.58904e... | Campaig... | P11_CAG... | 2001-10-... | 14.1 |
| 3 | 4 | P12 | 430859 | 4.58278e... | Campaig... | P12_CUP... | 2006-12-... | 95.08 |
| 4 | 5 | P2 | 428407 | 4.58149e... | Campaig... | P2_CPR1... | 2010-11-... | 0.6 |
| 5 | 6 | P3 | 432962 | 4.58464e... | Campaig... | P3_CCL0... | 2003-01-... | 57.9 |
| 6 | 6 | P3 | 432962 | 4.58464e... | Campaig... | P3_CUP0... | 2006-10-... | 84.22 |
| 7 | 6 | P3 | 432962 | 4.58464e... | Campaig... | P3-2013 | 2013-12-... | 42.18 |
| 8 | 7 | P4 | 431195 | 4.5814e... | Campaig... | P4_CPR1... | 2010-05-... | 0.36 |
| 9 | 7 | P4 | 431195 | 4.5814e... | Campaig... | P4-2013 | 2013-12-... | 12.86 |
| 10 | 8 | P5 | 429845 | 4.58109e... | Campaig... | P5_CCL0... | 2003-01-... | 70.3 |
| 11 | 8 | P5 | 429845 | 4.58109e... | Campaig... | P5_CPR1... | 2010-05-... | 0.0001 |
| 12 | 8 | P5 | 429845 | 4.58109e... | Campaig... | P5-2013 | 2013-12-... | 14.15 |
| 13 | 8 | P5 | 429845 | 4.58109e... | Campaig... | P5-2014 | 2014-01-... | 27.77 |
| 14 | 9 | P6 | 433176 | 4.5882e... | Campaig... | P6-2013 | 2013-12-... | 2.185 |

Map settings

Value to use

○ Earliest ○ Latest ● Average ○ Minimum ○ Maximum

*Figure 5.10 - Resulting maps of distribution the Nitrates (mg/l) classified according to a given normative using the Parameter Normative Map Tools.*

Further tools included in this module provide different query forms that enable the user to automatically query parameters such as major ions for the selected samples. In addition, different methodologies for exporting the results of such queries are available (*e.g.* .ods format) for being further processed or reported.

### 5.2.6 Hydrochemical external format

This set of tools enables to use different query criteria to be exported to external formats (e.g. .ods, .xls or .csv files). In addition, these commands allow to export the query data to external platforms (e.g. Easy_Quim, MIX or Statistical Tools). For more information about this set of free external platforms see: http://h2ogeo.upc.edu/en/investigation-hydrogeology/software

E *Export to Easy_Quim:* This free tool provides a query form that enables the user to automatically query the common major ions of the selected sample query, to export results to different formats (e.g. .xls, .csv, .ods formats), including the query of the major ions for the selected Query (i.e. for the selected points in the desired intervals). The resulting spreadsheet follows the required format of EasyQuim.

M *Export to Mix:* This free tool provides a query form that enables the user to apply different query criteria oriented to evaluate the mixing ratios of the samples. As result, a classification of the samples in end-members (extreme composition of a mixing) or in samples-members can be obtained for the selected parameters, that can be exported to different portable format (e.g. .xls, .csv, .ods format). The resulting spreadsheet follows the required format of Mix.

Q *Export to Statistical Analysis*: This free tool enables the user to obtain, for the selected query sample and for the selected parameters, different portable reports for further analysis (e.g. statistical analysis).

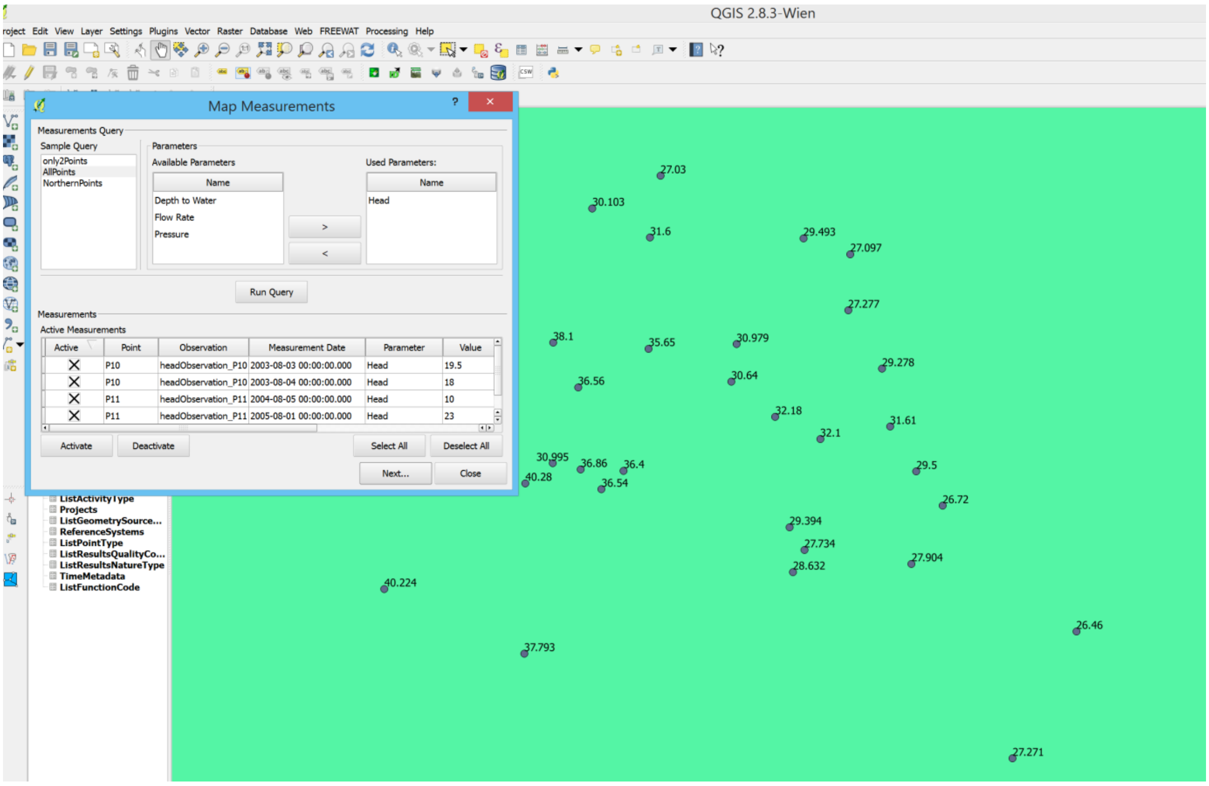
### 5.2.7 Hydrogeological Spatial Query

*Hydrogelogical Spatial Query:* This command aids to quickly check data related to hydrogeological measurements (e.g. head, wells abstractions, etc.) performed in wells, piezometers, springs, etc. This query only act for those points where hydrogeological observations and measurements have been introduced into the database.

Use this command to create and add database queries of the selected points (spatial selection) and for the desired time interval. Also campaign information can be used as query criteria. Thus, hydrogeological tools can be applied to queries stored in the database or it can be used in future analysis.

### 5.2.8 Hydrogeological Parameter Map

***Hydrogeological Parameter Map:*** This command enables to obtain general statistical parameters (minimum, maximum or mean value) of the selected group of data for being mapped. The results can be saved as .ods format. Fig. 5.11 shows a resulting map of piezometric head created with this tool. Further analysis (e.g. piezometric surfaces) can be done by using the inherent capabilities of QGIS.



Measurements Query

Sample Query: only2Points, AllPoints, NorthernPoints

Parameters

Available Parameters: Depth to Water, Flow Rate, Pressure

Used Parameters: Head

Run Query

Measurements

Active Measurements

| Active | Point | Observation | Measurement Date | Parameter | Value |
|---|---|---|---|---|---|
| ☒ | P10 | headObservation_P10 | 2003-08-03 00:00:00.000 | Head | 19.5 |
| ☒ | P10 | headObservation_P10 | 2003-08-04 00:00:00.000 | Head | 18 |
| ☒ | P11 | headObservation_P11 | 2004-08-05 00:00:00.000 | Head | 10 |
| ☒ | P11 | headObservation_P11 | 2005-08-01 00:00:00.000 | Head | 23 |

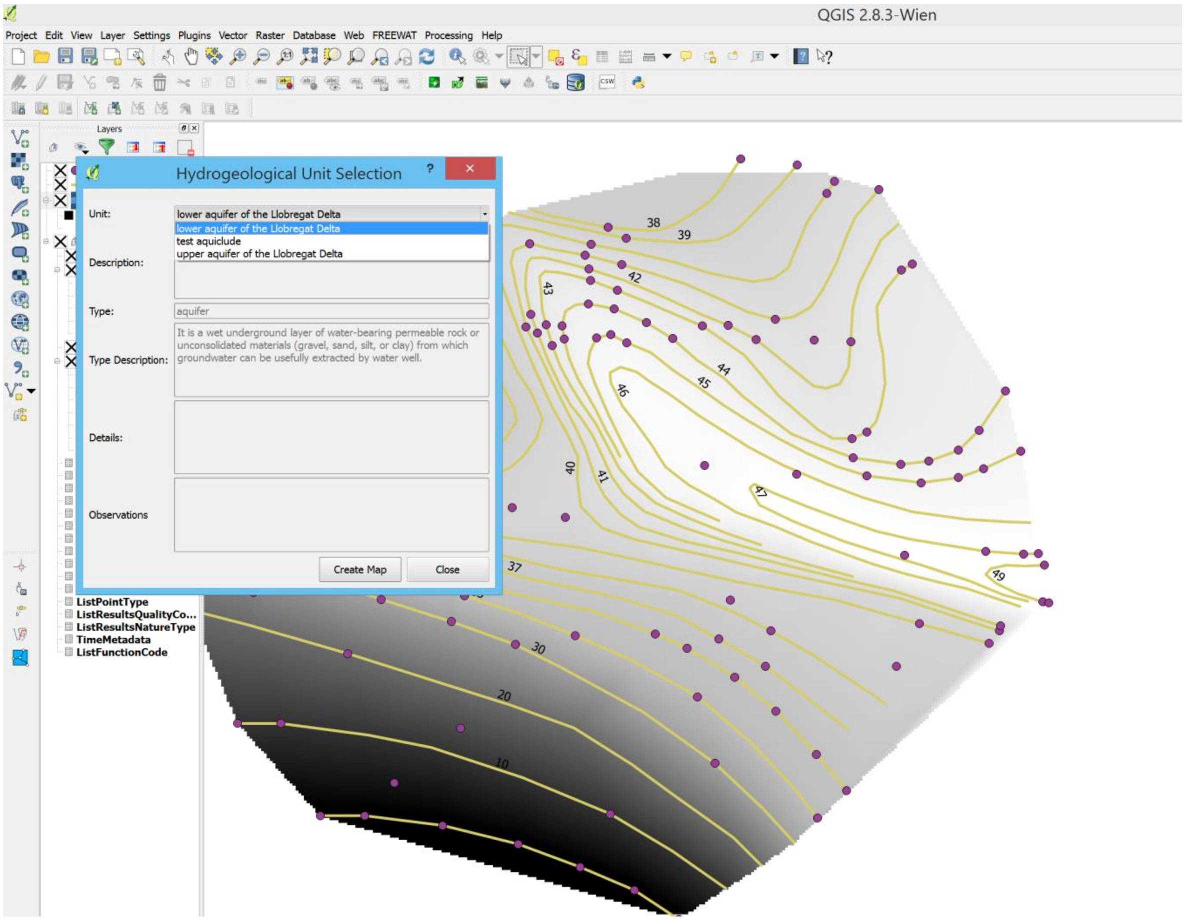
Activate Deactivate Select All Deselect All Next... Close

*Figure 5.11 - Sketch showing some results obtained using Hydrogeological Parameter Map Tools.*

### 5.2.9 Hydrogeological Units Map

***Hydrogeological Units Maps:*** This command enables the user to create maps of top/bottom for the selected hydrogeological units defined in the wells. First of all, the user chooses the hydrogeological unit that want to query, then a map showing top and/or bottom of each selected hydrogeological unit is performed as a result.

Fig. 5.12 shows the top of one hydrogeological Unit. To interpolate this information and made a hydrogeological surface, QGIS has spatial Interpolation options. Thus, this layer can be used to delimitate the Hydrogeological Unit geometry and apply it to build a numerical model with other FREEWAT tools.



Hydrogeological Unit Selection

Unit: lower aquifer of the Llobregat Delta

lower aquifer of the Llobregat Delta
test aquiclude
upper aquifer of the Llobregat Delta

Description:

Type: aquifer

Type Description: It is a wet underground layer of water-bearing permeable rock or unconsolidated materials (gravel, sand, silt, or clay) from which groundwater can be usefully extracted by water well.

Details:

Observations

*Figure 5.12 - Sketch representing the functionality of the Hydrogeological Units Maps tools. The user can create and export hydrogeological units for further analysis (e.g. to build a numerical model).*

### 5.2.10 Hydrogeological Time Plot

***Hydrogeological Time Plot***: To create time plots of selected parameters of a selected query (previously created with the *Hydrogeological Spatial Query* tool). Henceforward, the results are saved in a table (e.g. as .csv, .ods formats) and automatically a time plot is performed.

# 6 Observation Analysis Tool (OAT)

The Observations Analysis Tool is a tool to import, export and process time-series data, implemented within the FREEWAT GIS environment. OAT is a Python package which is integrated in the FREEWAT environment through an interface exposing its features to modelers and non-programmer users. Within the organization of the FREEWAT plugin (Fig. 1.1), OAT falls outside the core Hydrological module and fills the function of a tool for general operations to prepare input data and post-processing functionalities; for time series data generally, as well as online sensors specifically. Additionally, OAT provides a direct link to the observation creation portion of the sensitivity analysis/calibration block. Groundwater head and river flux observations can be read directly from OAT sensors. In FREEWAT, the observations can be loaded into QGIS as CSV files directly and specified through the Create Head Observation Layer interface. Within this interface is a checkbox which can be used to activate the link to a selected OAT sensor. Provided the sensor has the correct attributes, and metadata, it will be used for the head observations. The same can be done within the Create Flow Observation layer interface. In this case, the difference between two sensors is used, i.e. the difference between two streamflow gages representing the water gained or loosed by a river segment between the two gages.

## 6.1 OAT conceptualisation and FREEWAT implementation

The OAT library implements two main classes: the Sensor class that is designated to handle time-series data and metadata and the Method class which is designated to represent a processing method. The library applies the behavioral visitor pattern which allows the separation of an algorithm from the object on which it operates: thanks to this design pattern it is possible to add a new processing capability by simply extending the Method class without the need to modify the Sensor class. From a dependency point of view, OAT takes advantage of the PANDAS, NUMPY and SCIPY packages.

The Sensor class took inspiration from the Sensor Observation Service standard and it's istSOS implementation (Cannata & Antonovic, 2010), with a specific simplification to ease its usage. istSOS (Istituto Scienze della Terra Sensor Observation Service) has been used since 2009 as a simple implementation of the SOS for the management, provision and integration of hydro-meteorological data collected in Canton Ticino. It has also been used with the Verbano Lake Early Warning System for lake level forecasting and flooding hazard assessment (Cannata et. al., 2015). istSOS has also been used to develop a smart system to sustain optimal water usage in irrigation within the European project ENORASIS (Cannata & Antonovic, 2015).

The second major component of OAT is its processing capabilities, based initially on the open source TSPROC (Time Series Procressing) developed by Westenbroek et al. (2012). TSPROC is model-independant but is often used with PEST (Doherty 2004) as a pre- and postprocessor during model optimisation. TSPROC has been used in with a wide range of codes for hydrologic models (Kim et. al, 2014; Cocca et. al., 2003; Skahill et. al., 2012), to "distill the large time-series data set to a smaller set of observations and summary statistics that captured the salient hydrologic information" for both model input and output (Walker et. al. 2007).

The general structure and implemented usage of the library in FREEWAT is depicted in Fig. 6.1.

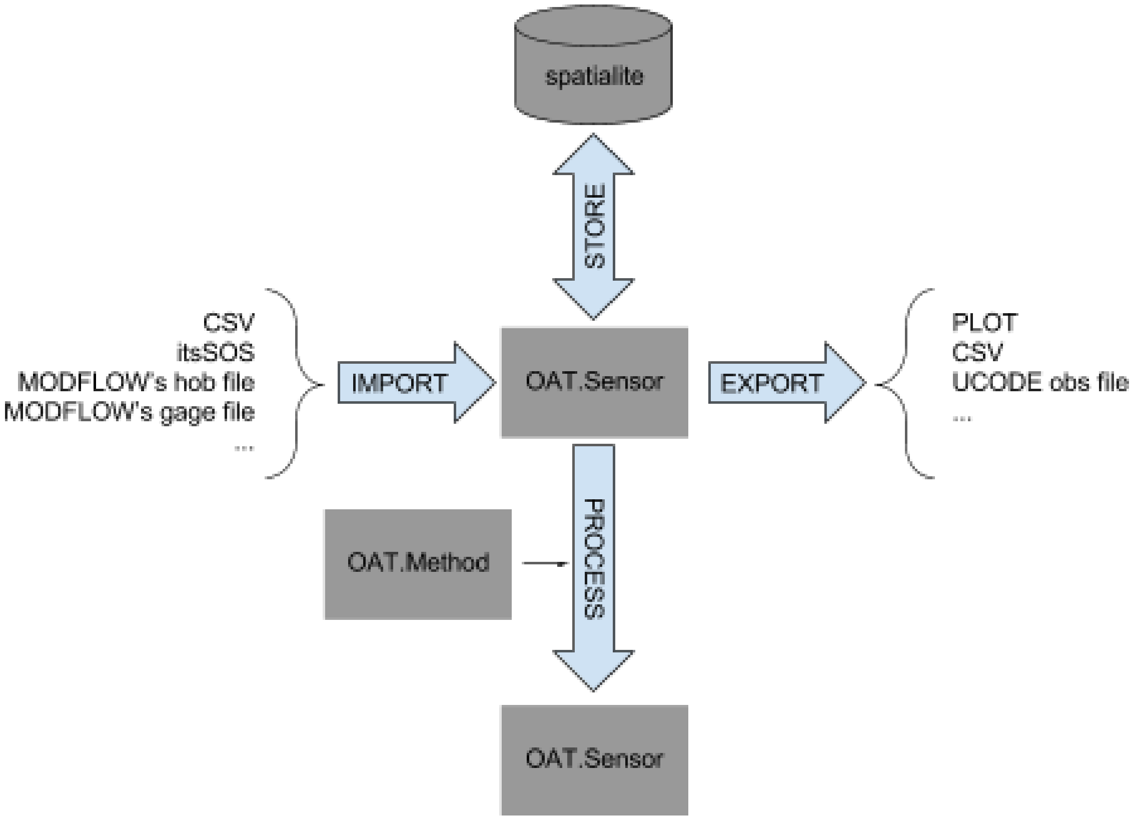
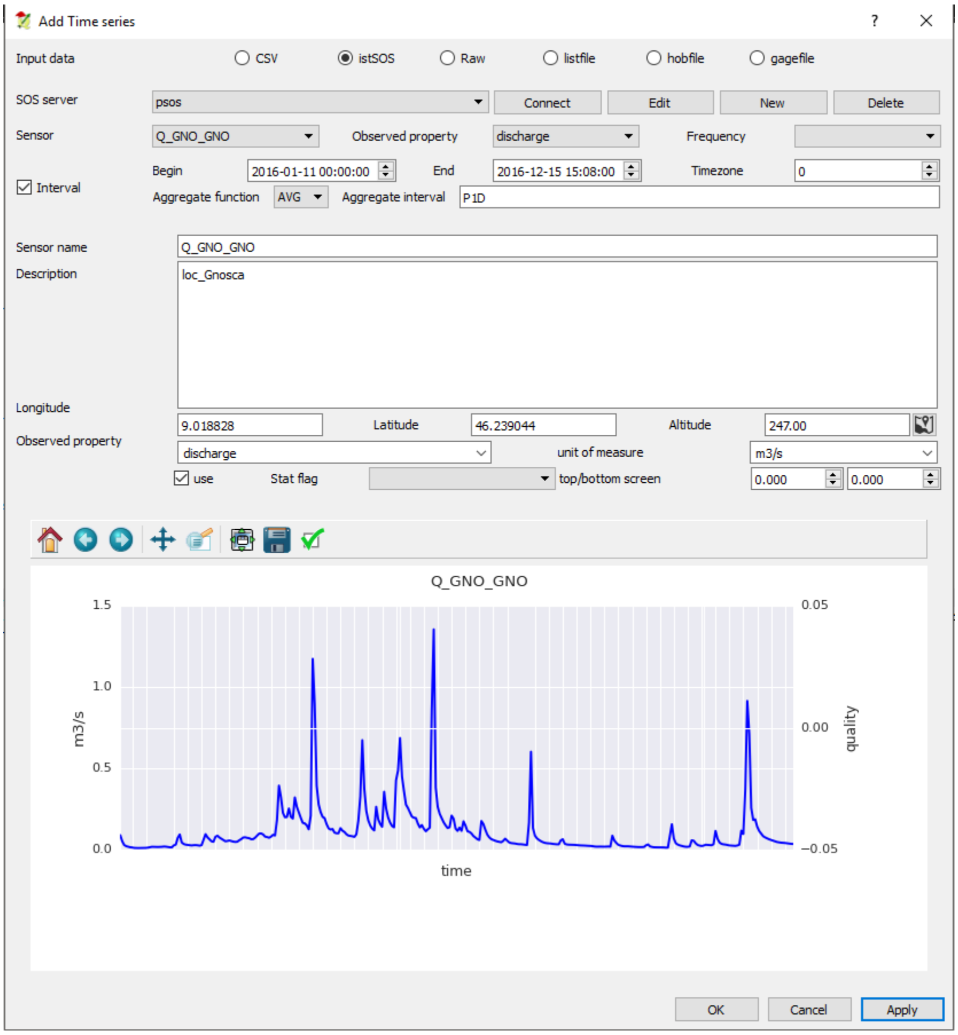


*Figure 6.1 - The OAT data retrieval, processing, storage and export workflow.*

Each Sensor object is characterized by a single time-series that is represented by a data section consisting in a PANDA's time-series and a location/metadata section. Every OAT.Sensor object can be stored in a spatialite DB and re-loaded back in OAT.Sensor with its own data and metadata. For each sensor, the "metadata section" includes name, description, location (lat, lon, elev), unit of measure, observed property, coordinate system, time-zone, frequency, weight statistic and data availability (time interval). The "data section" contains time, data, name, and quality index, as well as a tag marking whether or not an individual observation in the series will be used. In addition to requesting sensor data from an istSOS server (Fig. 6.2), OAT can retrieve data stored in local files or databases into the FREEWAT GIS environment for further use. Further input accepted format may be implemented in the future. Additionally, model results can also be imported as OAT sensor for further time series analysis.



Input data ○ CSV ● istSOS ○ Raw ○ listfile ○ hobfile ○ gagefile

SOS server: psos

Sensor: Q_GNO_GNO

Observed property: discharge

Frequency:

☑ Interval

Begin: 2016-01-11 00:00:00

End: 2016-12-15 15:08:00

Timezone: 0

Aggregate function: AVG

Aggregate interval: P1D

Sensor name: Q_GNO_GNO

Description: loc_Gnosca

Longitude: 9.018828

Latitude: 46.239044

Altitude: 247.00

Observed property: discharge

unit of measure: m3/s

☑ use

Stat flag:

top/bottom screen: 0.000 0.000

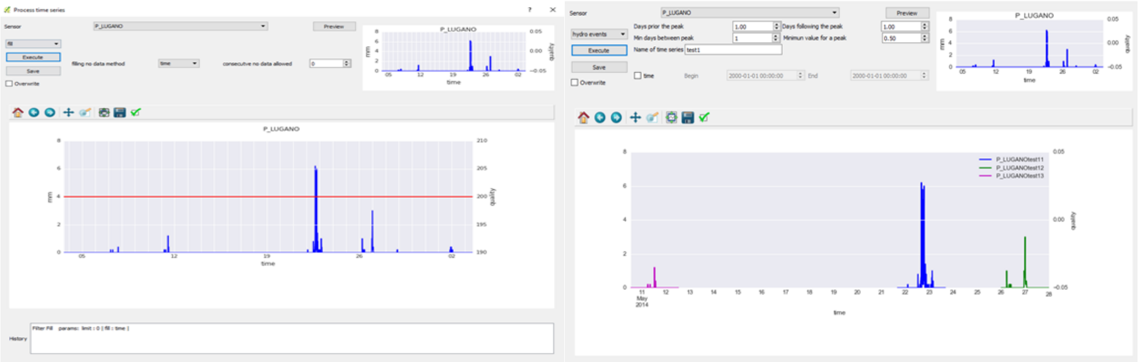
*Figure 6.2 - The interface for adding a new sensor from an istSOS server. The first row, "Input Data" shows the other possible sources of sensor data.*

The currently available OAT.Method objects are based on TSPROC processing capabilities (Westenbroek et al. 2012), They are presented in Tab. 6.1, with the addition of FREEWAT specific processes, and a couple of examples are shown in Fig. 6.3.

| Method | Effect |
|---|---|
| Digital Filter | The digital filter Method calculates a new Sensor by passing an existing Sensor through a digital filter to remove high or low frequency components. |
| Exceedence | The Exceedance probability method calculates the exceedance probability after Searcy (1959) for a particular Sensor. |
| Hydro events | The extract hydrologic events method can be used to extract hydrographs for a time period (in days) preceding and following a peak hydrologic event, such as a storm. |
| Hydro Indices | The hydrologic indices are a series of statistical measures of streamflow used to describe various ecologically important aspects of flow regime. |

|  |  |
|---|---|
| Quality | The set quality method can be used to set uniform quality values for a Sensor within time bounds, or to remove values that fall outside of specified value bounds. |
| Resample | The resample method calculates a new time-series with a given frequency by sampling values of a time-series with a different frequency. |
| Data values | The data values method can be used to set uniform data values for a Sensor within time bounds, or to remove values that fall outside of specified value bounds. |
| Hydrograph separation | The hydrograph separation method is a filter which produces two time-series, storm-flow and baseflow hydrograph, from stream discharge. |
| Integrate | Several methods are available for integrating a Sensors time-series with respect to time. |
| Compare | The Compare Sensors Method can be used to calculate several "Goodness of Fit" statistics for two Sensors |
| Subtract | The subtract method can be used to subtract the data values of a Sensor from the data values of the selected Sensor. |
| Fill | If a Sensor is missing data in the form of gaps, or if it contains no-data values, it can be filled using a variety of methods. |
| Statistics | The calculate statistics method returns some basic statistics for the Sensor time-series. |
| Hargreaves | The Hargreaves-Samani equation (Allen et. al. 1998) can be used to estimate the reference crop evapotranspiration using minimum climatological data. |

*Table 6.1 - The currently available methods available in OAT with a brief description. Full descriptions are presented in the FREEWAT Manual, Volume 5.*


Sensor: P_LUGANO
fill
filling no data method: time
consecutive no data allowed: 0
☐ Overwrite
History: Filter Fill   params:  limit : 0 | fill : time |

Sensor: P_LUGANO
hydro events
Days prior the peak: 1.00
Days following the peak: 1.00
Min days between peak: 1
Minimun value for a peak: 0.50
Name of time series: test1
☐ time   Begin 2000-01-01 00:00:00   End 2000-01-01 00:00:00
☐ Overwrite

*Figure 6.3 - The fill (left) and hydro events (right) processes in OAT, used on an example precipitation data set.*

The result of a process is generally a new OAT.Sensor, so that processes can be concatenated, and the final resulting time-series can be saved in the FREEWAT model database or exported. As discussed in the package design, this library of processes is open and expandable to incorporate users needs, specifically as an aspect of the participatory approach adopted in FREEWAT development.

In preparation for model input the raw data processing capability of OAT are particular important. In fact, since often available data will not have the same temporal discretisation as the chosen modelling time, various filter and resampling or data fill methods are available to create new, but statistically sound, time-series to be used as model input.

OAT incorporates the post-processing of model results by creating sensors with appropriate temporal discretisation for the visualisation and further processing of model results as time-series. MODFLOW head and flow observations, listing file volumetric budget components (either as cumulative budgets or rates), and gage file components are currently supported formats. Time-series visualisation is achieved through the Python Matplotlib module. Various elements of the volumetric water budget for a MODFLOW model can be loaded as sensors and visualised and compared, to other elements of the budget or to other relevant time-series or observations (Fig. 6.4).

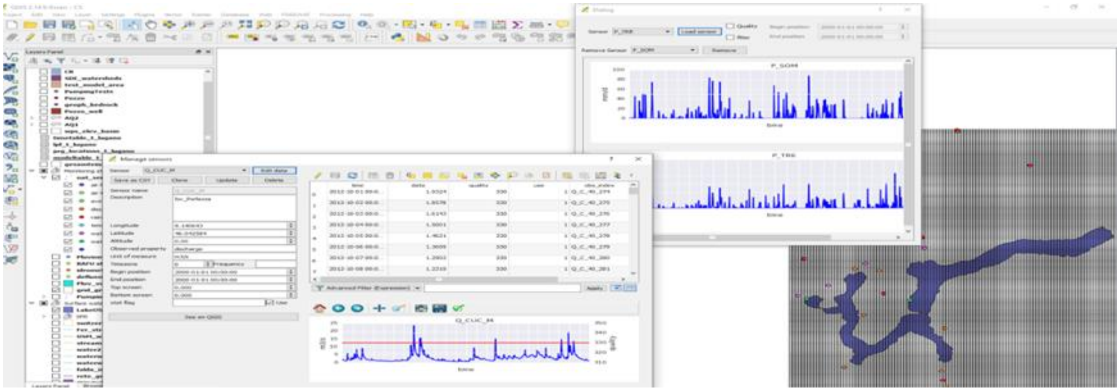


*Figure 6.4 - Screenshot of the OAT menu in a sample simulation within QGIS showing the Manage sensor and Compare sensor interfaces, with the case study grid in the background.*

Groundwater head observations can be incorporated into the standard MODFLOW head observation package using OAT and simulated heads can be uploaded from the model results. Specifically for transient models with longer and possibly non continuous observation periods, OAT offers improved visualization capabilities because the observations are separated by well and correctly plotted along the time axis so that is possible to true to time with variable spacing along the x-axis, rather than just sequentially, which allows their comparison with other time-series, e.g. nearby GW stresses.

# 7 Calibration and Sensitivity module

UCODE_2014 is a universal software tool designed to perform sensitivity analysis, uncertainty analysis, and model calibration (parameter estimation), as well as provide auxiliary tools for analyzing data worth and data needs assessment. UCODE_2014 is designed to work with text input and output files, and therefore, can interact with simulation models in a generic way. UCODE_2014 automates the process of adjusting model parameter values, simulating the models, and examining the simulated results (the procedure often done by hand). By automating this often arduous process, and providing a common framework to compare both models and parameters, and their uncertainties, the end result should be better and more transparent models.

Model calibration is a process whereby various aspects of process-model construction and parameter values are modified in hopes of improving how well the model represents the system of concern. Part of model calibration includes changing the values of selected parameters used to define model inputs. This part of the model calibration process is amenable to application of optimization. This Parameter estimation is the process of adjusting model parameters, within reasonable limits, such that simulated equivalents are as close as possible to observed values.

Numeric model outputs include simulated values that may be compared to measured (or “observed”) values and simulated values that may be considered predictions of future conditions. Observed values commonly are referred to as “observations,” and the corresponding model-simulated values are referred to as “simulated equivalents to observations.”

How can UCODE_2014 assist in evaluating our models and our model results? Sensitivity analysis and calibration can provide answers to the following questions:

1. What parameters are supported by the observations?
2. Are any of the parameters dominated by a single observation?
3. What model parameters are important to the things I need to predict?
4. What data should be collected to improve the predictions?
5. Which conceptual model of the system is likely to produce better predictions?
6. How certain are the predictions?

As graphically explained in Fig. 7.1, the calibration and sensitivity analysis module is directly connected to the OAT module (using OAT it is possible to create the observation files for UCODE as mentioned later) and is then embracing and interacting with all the other modules. Once the main physical model will be developed, then the workflow would ideally go from using the observation files created by OAT and then start the model sensitivity analysis and calibration using UCODE. Then, it is expected that UCODE will provide suggestions on how to improve the physical model and therefore the entire process can be developed again until the model results are satisfactory.

With UCODE integrated in FREEWAT, the UCODE input file, template file, and instruction files (Fig. 7.1) are automatically created based on a selection of parameters and observations. In the current implementation, UCODE_2014 can be run in three modes: 1) forward (basic model simulation), 2) sensitivity analysis, or 3) parameter estimation/optimization.

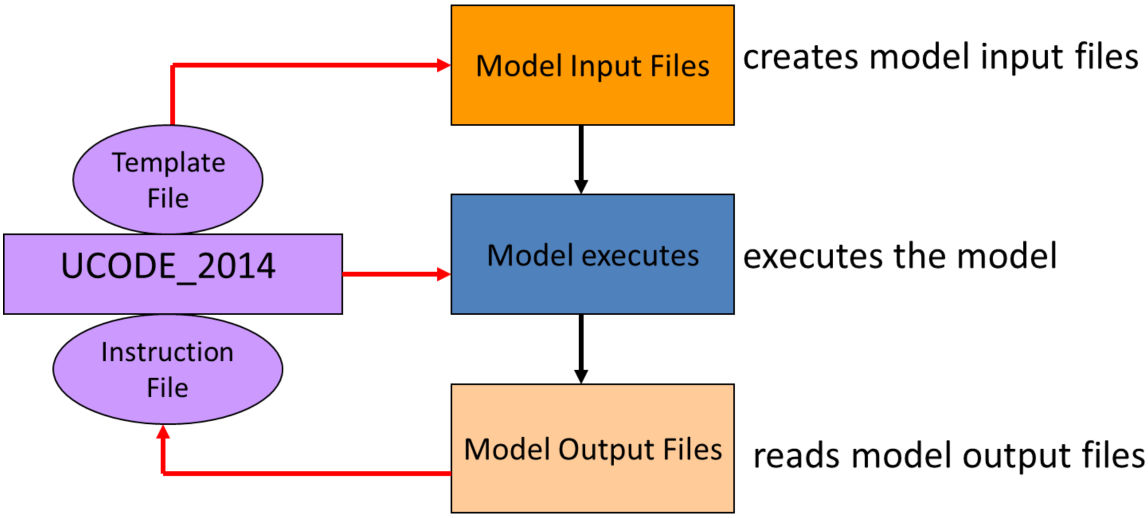


*Figure 7.1 – Schematic of UCODE operation and of the files needed to run the software.*

As mentioned above, UCODE_2014 is based on local sensitivity analysis and supports the use of error-based weighting in model development.

Error-based weighting has appeared in the literature extensively (for example, Hill, 1992; Hill, 1998; Mroczkowski et al., 1997; Oliver et al., 2008; Foglia et al., 2009, Renard et al., 2011).

While local sensitivity analysis does not capture some aspects of nonlinearity that can be captured by the other methods, such as local minima, experience has demonstrated that many models of natural systems are linear enough for local sensitivity analysis methods to be useful (for example, groundwater flow and advective transport in work by Anderman et al. [1996], Poeter and Hill [1997], Hill et al. [1998], D'Agnese et al. [1999], Yager [1998, 2004], Hill [2006], and Foglia et al. [2007]; conservative and reactive groundwater transport in work by Barlebo et al. [1998, 2004], Mehl and Hill [2001], Barth and Hill [2005a, 2005b], and Hill et al. [2006]; and streamflow and transport in work by Scott et al. [2003] and Gooseff et al. [2005]; Hill and Tiedeman [2007] provide an overview).

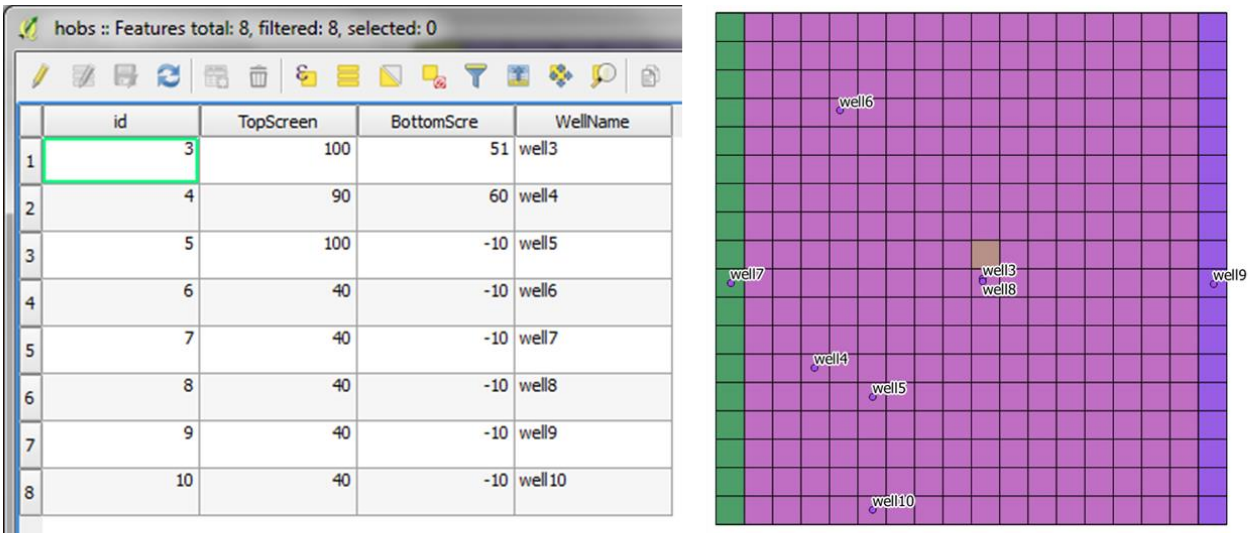
## 7.1 Structure of the calibration and sensitivity package

### 7.1.1 Creating observation file

All types of observations are currently supported in FREEWAT: hydraulic head observations using the HOB Package of MODFLOW and flow gain/loss observations from constant head, drain, general head, or river boundaries, using the CHOB, DROB, GBOB, or RVOB Packages of MODFLOW, respectively. These observations can also be derived from the Observation Analysis Tool (OAT) included with FREEWAT.

### 7.1.2 Head Observation File

Fig. 7.2 shows an example of the observation file created for including head observations. If available, it allows the user to include information on the well screens and well depth. Each well needs to be identified by a well name that will be used both in MODFLOW and UCODE_2014.



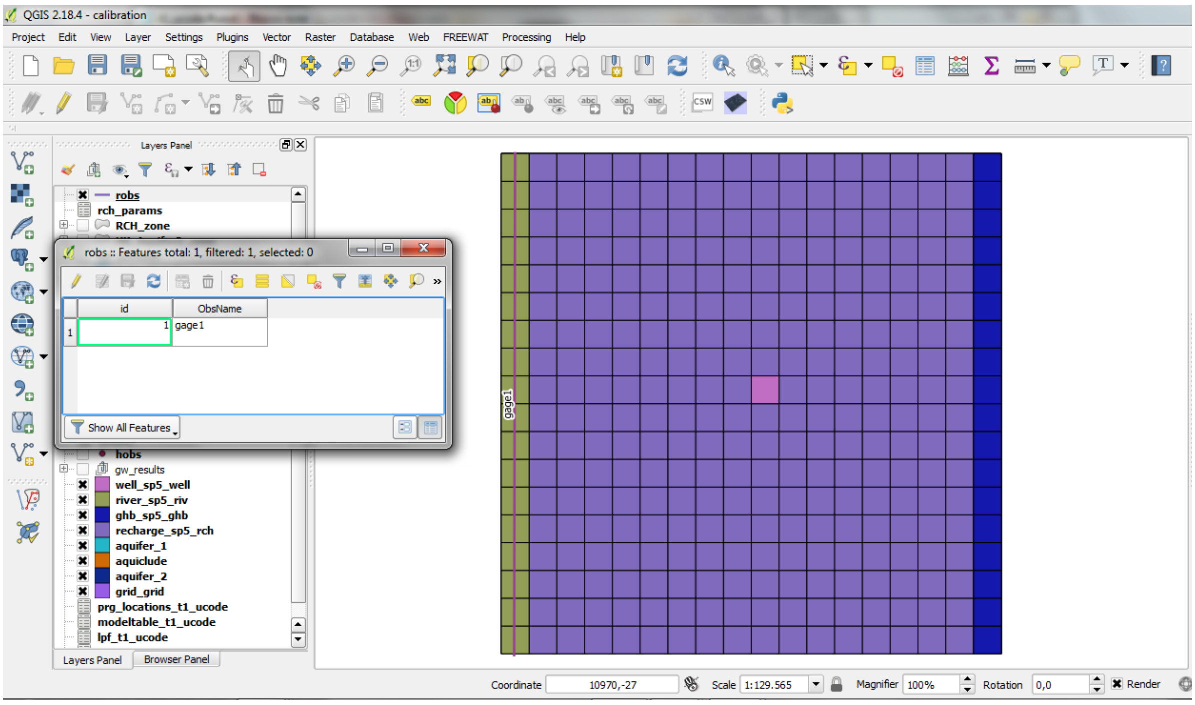
hobs :: Features total: 8, filtered: 8, selected: 0

|   | id | TopScreen | BottomScre | WellName |
|---|---|---|---|---|
| 1 | 3 | 100 | 51 | well3 |
| 2 | 4 | 90 | 60 | well4 |
| 3 | 5 | 100 | -10 | well5 |
| 4 | 6 | 40 | -10 | well6 |
| 5 | 7 | 40 | -10 | well7 |
| 6 | 8 | 40 | -10 | well8 |
| 7 | 9 | 40 | -10 | well9 |
| 8 | 10 | 40 | -10 | well10 |

*Figure 7.2 – Example of Head Observation File.*

### 7.1.3 Flow Observation File

Fig. 7.3 shows an example for the flow observation file. It is for the RIV package and creates gains/losses observations (RVOB). Gain/loss observations are created selecting the all reach where the fluxes between river and aquifer are calculated and the observation will be the total net flow between river and aquifer.



robs :: Features total: 1, filtered: 1, selected: 0

|   | id | ObsName |
|---|---|---|
| 1 | 1 | gage1 |

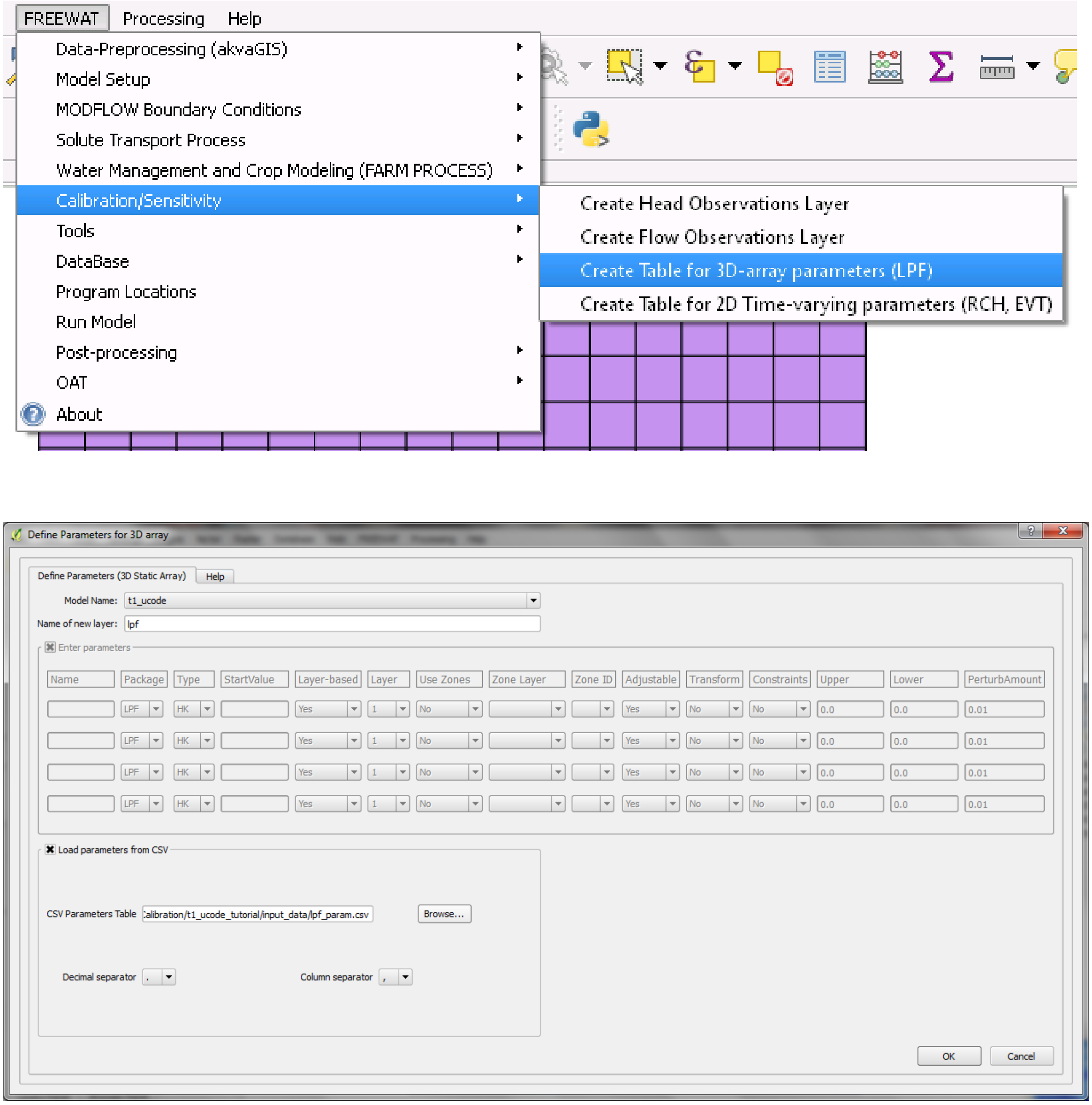
Coordinate 10970,-27 Scale 1:129.565 Magnifier 100% Rotation 0,0

*Figure 7.3 - Example of river observation file (RVOB).*

## 7.1.4 Selecting parameters

There are two possible types of parameters which can be included 3D parameters and 2D Time-variant parameters. Parameters related to the three dimensional array inputs of the LPF (Layer Property File) Package of MODFLOW are available, such as hydraulic conductivity (HK), vertical hydraulic conductivity (VKA), specific storage (SS), and specific yield (SY) are fully integrated in the platform. As parameters related to the two dimensional time-variant array inputs of the RCH (Recharge) and EVT (Evapotranspiration) Package of MODFLOW are available. The parameters can apply as a constant multiplier for the entire model layer (by designating the “layer based” option) or can apply to a specific zone of the model (as designated by a zone ID and associated QGIS layer of zonal values). This allows flexibility in defining the parameterization that represents the heterogeneity of the aquifer system. Existing parameterizations can be loaded in from a CSV file to allow users to quickly compare different parameterizations. Options for parameter estimation and (like log transformation and the maximum allow parameter change) as also supported.

Fig. 7.4 shows the details of how the parameter capability has been added into FREEWAT and which type of information are needed to create the parameters template files.


FREEWAT
Processing
Help
Data-Preprocessing (akvaGIS)
Model Setup
MODFLOW Boundary Conditions
Solute Transport Process
Water Management and Crop Modeling (FARM PROCESS)
Calibration/Sensitivity
Tools
DataBase
Program Locations
Run Model
Post-processing
OAT
About
Create Head Observations Layer
Create Flow Observations Layer
Create Table for 3D-array parameters (LPF)
Create Table for 2D Time-varying parameters (RCH, EVT)
Define Parameters for 3D array
Define Parameters (3D Static Array)
Help
Browse...
OK
Cancel

Model Name: t1_ucode

Name of new layer: lpf

☒ Enter parameters

| Name | Package | Type | StartValue | Layer-based | Layer | Use Zones | Zone Layer | Zone ID | Adjustable | Transform | Constraints | Upper | Lower | PerturbAmount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | LPF | HK |  | Yes | 1 | No |  |  | Yes | No | No | 0.0 | 0.0 | 0.01 |
|  | LPF | HK |  | Yes | 1 | No |  |  | Yes | No | No | 0.0 | 0.0 | 0.01 |
|  | LPF | HK |  | Yes | 1 | No |  |  | Yes | No | No | 0.0 | 0.0 | 0.01 |
|  | LPF | HK |  | Yes | 1 | No |  |  | Yes | No | No | 0.0 | 0.0 | 0.01 |

☒ Load parameters from CSV

CSV Parameters Table :alibration/t1_ucode_tutorial/input_data/lpf_param.csv

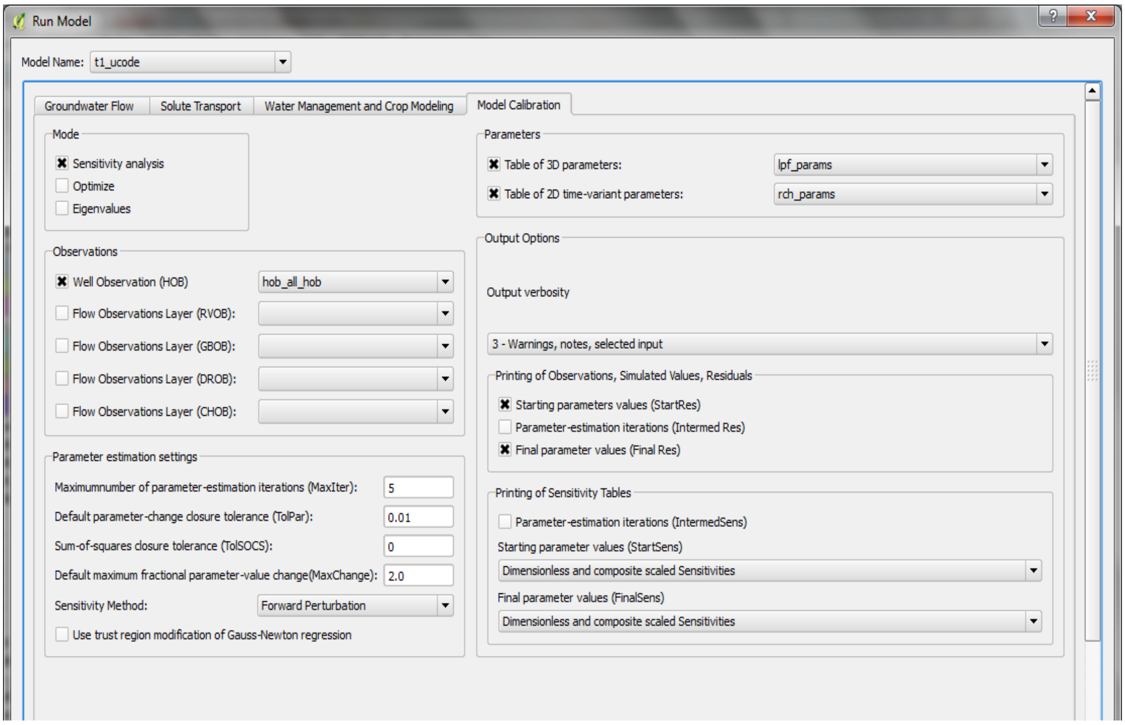
Decimal separator .

Column separator ,

*Figure 7.4 – Interface for building the LPF parameters template file.*

### 7.1.5 Model sensitivity analysis and calibration and post-processing of results

Sensitivity analysis and calibration capabilities have been developed and tested using different types of observations, as mentioned above. For some of the parameters needed to run UCODE_2014, standard values have been already wired into the code, but the user is always able to define different parameter values if needed. These UCODE_2014 running parameters include for example the number of iterations, the size of parameters perturbation, closure criteria. Fig. 7.5 shows an example of the UCODE_2014 running window.



Run Model

Model Name: t1_ucode

Groundwater Flow | Solute Transport | Water Management and Crop Modeling | Model Calibration

Mode

☒ Sensitivity analysis
□ Optimize
□ Eigenvalues

Parameters

☒ Table of 3D parameters: lpf_params
☒ Table of 2D time-variant parameters: rch_params

Observations

☒ Well Observation (HOB) hob_all_hob
□ Flow Observations Layer (RVOB):
□ Flow Observations Layer (GBOB):
□ Flow Observations Layer (DROB):
□ Flow Observations Layer (CHOB):

Output Options

Output verbosity

3 - Warnings, notes, selected input

Printing of Observations, Simulated Values, Residuals

☒ Starting parameters values (StartRes)
□ Parameter-estimation iterations (Intermed Res)
☒ Final parameter values (Final Res)

Parameter estimation settings

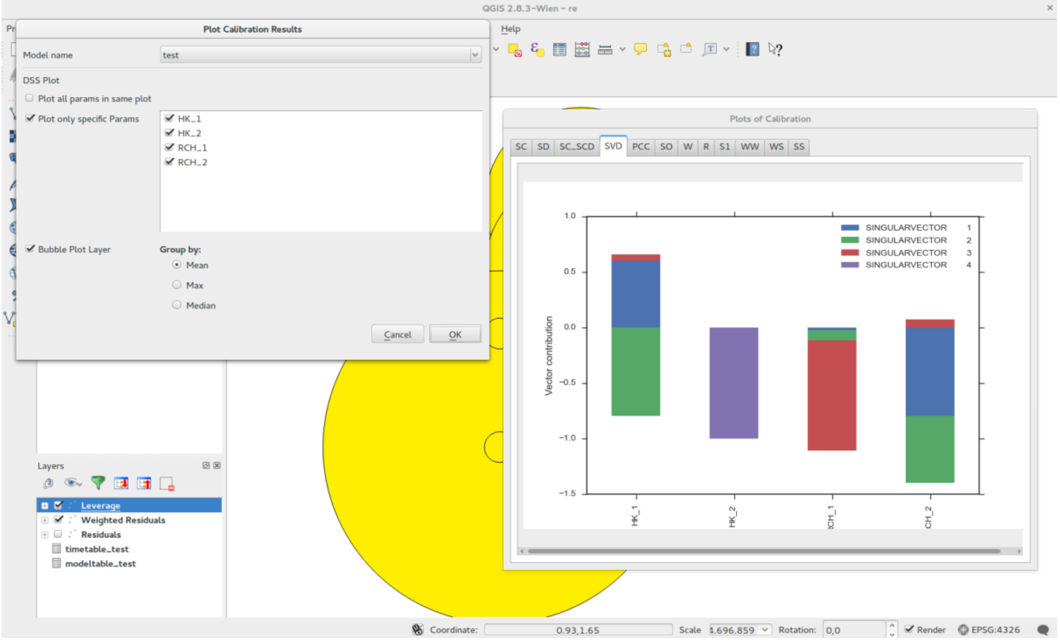
Maximumnumber of parameter-estimation iterations (MaxIter): 5
Default parameter-change closure tolerance (TolPar): 0.01
Sum-of-squares closure tolerance (TolSOCS): 0
Default maximum fractional parameter-value change(MaxChange): 2.0
Sensitivity Method: Forward Perturbation
□ Use trust region modification of Gauss-Newton regression

Printing of Sensitivity Tables

□ Parameter-estimation iterations (IntermedSens)
Starting parameter values (StartSens)
Dimensionless and composite scaled Sensitivities
Final parameter values (FinalSens)
Dimensionless and composite scaled Sensitivities

*Figure 7.5 – UCODE_2014 running.*

When performing sensitivity analysis, it is often most effective to plot measures of parameter and/or observation importance to quickly determine key relationships between model parameters and observed data. For these reasons, development of plotting capabilities is currently being developed and tested. Fig. 7.6 presents an example of the plotting capabilities under development.



QGIS 2.8.3-Wien - re

Plot Calibration Results

Model name: test

DSS Plot

□ Plot all params in same plot
☑ Plot only specific Params
☑ HK_1
☑ HK_2
☑ RCH_1
☑ RCH_2

☑ Bubble Plot Layer

Group by:
● Mean
○ Max
○ Median

Layers

☑ Leverage
☑ Weighted Residuals
□ Residuals
timetable_test
modeltable_test

Coordinate: 0.93,1.65 Scale 1.696.859 Rotation: 0,0 ✓ Render EPSG:4326

*Figure 7.6 – Example of post-processing plots.*

# 8 Examples of FREEWAT application

## 8.1 Application of Lake (LAK) package

The lake package as a part of FREEWAT will be demonstrated through the case study of SUPSI-IST. The Lugano Lake case study investigates the groundwater surface water interactions between the Lugano Lake and the main porous aquifers that are connected to the lake basins laterally. Although usually used to simulate seepage lakes within porous sandy aquifers, the Lugano case study will apply the lake package in an alpine lake setting where large sections of the lake are not in contact with the aquifers, only with bedrock, which is currently not simulated (Fig. 8.1). The streams and rivers, within the simulated main aquifers, will be simulated using the SFR, as will the connections between the lakes. The Lugano lake will be modelled as three separate lakes. The northern basin, connected to the southern basin with a stream segment 50 m wide, and the southern basin connected to the small Ponte Tresa basin, just before the outflow from the watershed, with another 50 m wide stream segment. Additionally, the small north-eastern Lago di Piano near Porlezza will also be simulated. The main concern of stakeholders in the area is the eutrophication due to phosphorous load to the lake. Although phosphorus is only conditionally mobile in groundwater, the case study will investigate if those conditions are met, i.e. if the flow within the simulated aquifers is primarily from the streams to the aquifers.

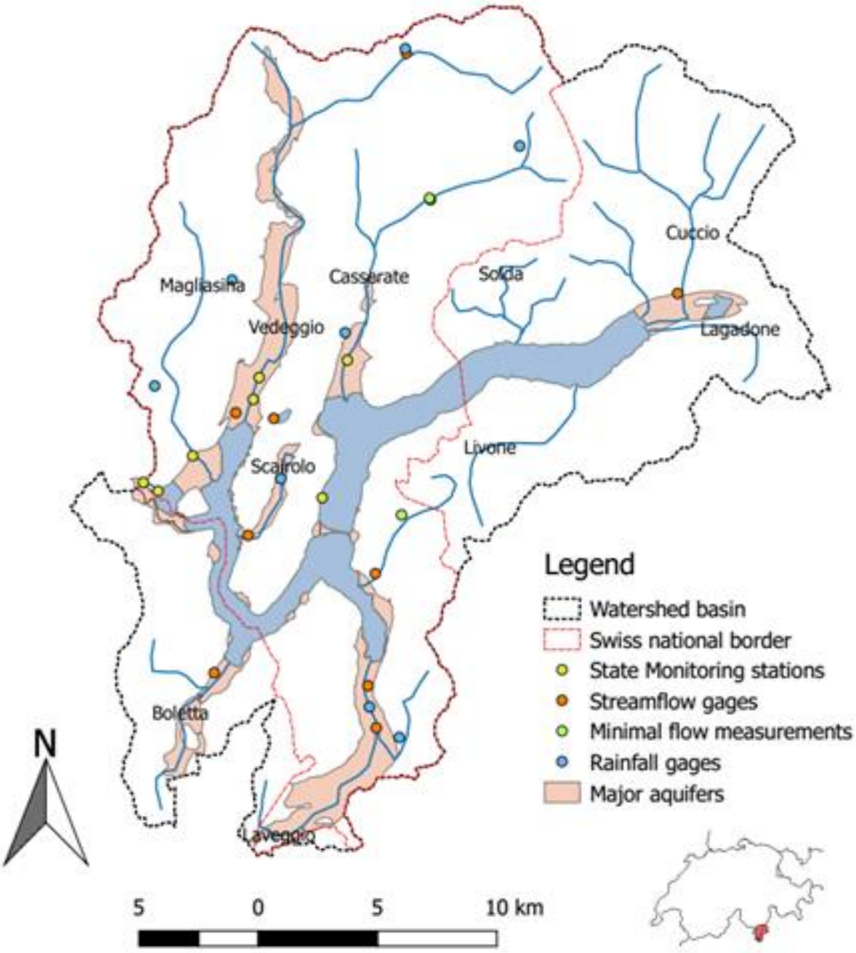


*Figure 8.1 - The lake Lugano case study area.*

The model will also consider groundwater abstraction within the major aquifers and recharge from precipitation. To date, the model grid has a horizontal discretization of 200 by 200 meters. Vertical discretization is currently two model layers with irregular depth, dependent on the DEM, geology and lake bathymetry. The lake is contained within the upper layer. Although subject to change, the current model time is 24 monthly stress periods with daily time steps from 1st Oct. 2012 to 30th Sept. 2014.

## 8.2 Application of water management and crop growth modeling

Application of MODFLOW-OHWM has been tested in several water management models, as in the following.

- Hansons et al. (2010) applied MODFLOW-OHWM to model micro-agriculture system in the Pajaro Valley (USA). The Authors analyzed the simulation of an aquifer storage and recovery system and related coastal water distribution system, in response to climate variations and additional supplemental sources such as local runoff. Such application demonstrated the effectiveness of applying MODFLOW-OWHM for simulating natural and anthropogenic components of the hydrologic cycle, aiming at analyzing and managing the distribution and dynamics of water supply and demand.
- Furthermore, in Faunt (2009) Authors reported the application of MODFLOW-OWHM to a larger and more complicated system extending over the Central Valley in California (USA). The latter covers about 20.000 square miles and it is one of the most productive agricultural regions in the world. Here, irrigated agriculture heavily relies on surface-water diversions and groundwater abstraction. Approximately one-sixth of the Country's irrigated land is in the Central Valley, and about one-fifth of the Country's groundwater demand is supplied from its aquifers. The Central Valley also is rapidly becoming an important area for California's expanding urban population. This surge in population has increased the competition for water resources within the Central Valley and statewide, which likely will be exacerbated by anticipated reductions in deliveries of Colorado River water to southern California. In response to this competition for water, a number of water-related issues have gained prominence: conservation of agricultural land, conjunctive use, artificial recharge, hydrologic implications of land-use change, and effects of climate variability. To analyze such a complex water competition and to help decision making, the U.S. Geological Survey's (USGS) Groundwater Resources Program is conducting a large-scale multidisciplinary regional study on groundwater availability, in the Central Valley Aquifer System, coupling the use of long-term groundwater monitoring data with groundwater models: simulations incorporate time-varying stresses and they can be used to evaluate the effects of both climatic and anthropogenic temporal changes in recharge and discharge on the hydrologic system between October 1961 and September 2003. This study is a valuable example of application of a numerical model as a decision support system: a continuous update of data will allow a periodic re-assessment of model setting and an up-to-date calibration of the model itself, eventually leading to a dynamic tool for effective water management.

Such examples show also how MODFLOW-OWHM can consider river ecological status and nature conservation, introducing constraints to take into account minimum environmental flow and sustainable hydrologic regime, in addition to withdrawal rules and demand side management options.

The following synthetic test aims at showing the successful application of FREEWAT as a professional tool to address water management issues in both urban and rural areas, leading to a shared and conjunctive use of ground- and surface-water resources. This example is inspired to a hypothetical case study presented in Schmid et al. (2006). The model domain is set in an alluvial valley slightly sloping from west to east.

The domain includes two water units (namely an agriculture farm and an urban zone), with three types of crops/vegetation (urban vegetation, orchard and pasture) and three soil types (silt, silty clay, and sandy loam). No crop rotation or fallow periods are simulated in this modeling exercise. One single hydro stratigraphic unit having a flat basal surface (0 m a.m.s.l.) is identified: a sandy unit, with variable thickness between 277 m and 300 m, hosting a porous, phreatic aquifer. The domain is crossed by a main surface canal delivering water to the irrigation district through an irrigation canal and by means of a gauge. As a further source of freshwater, groundwater is pumped through ten vertical wells, 4 out of which deliver water to the irrigation district, while the remaining are devoted to satisfy water demand of the urban area (Fig. 8.2). The simulation is run in transient conditions over one year (365 days).

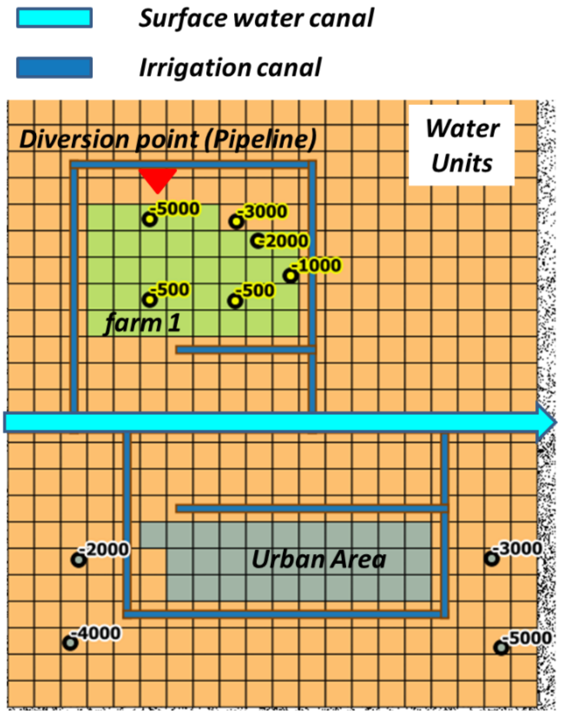


*Figure 8.2 - Set up of the synthetic model used to illustrate the application of water management module in rural and urban areas.*

The purpose of this model is to couple the MODFLOW model with optimization strategies aimed at satisfying the water demand raised by the two water units (rural and urban), resting on the same catchment.  In particular, the specific objectives of the modeling study are:

1. To investigate the relationships between ground- and surface-water in terms of water budget.
2. To investigate the possibility (or not) to fully satisfy water needs of both farms, assuming that groundwater is the only water source available within the catchment.
3. To assess the (positive) effect of using conjunctively ground- and surface-water at the agriculture water unit, in order to prevent unbalanced depletion of the underground resources.

As a conclusion, the following could be stated about the application of MODFLOW-OHWM, through FREEWAT, to such simple modeling exercise:

- For the urban area, the groundwater pumping (which is the only source available) does not fully satisfy the water demand at certain periods (specifically between April and June), thus resulting in a deficit scenario.
- For the rural area, the water supply provided by groundwater pumping through farm wells fully satisfies the water demand, providing that groundwater is the only available source of freshawater.
- Activating also a point of diversion from the main surface canal towards the irrigation district allows to fully satisfy irrigation requirements of the rural area itself. In such case, no groundwater abstraction is needed anymore, thus leading to a more sustainable exploitation of the aquifer.

Therefore, this simple example shows as different scenarios of water management can be simulated within FREEWAT, considering a common reference model, and including/excluding water sources and/or water rights for each water units defined in the model domain.

## 8.3 Application of sensitivity and calibration module

UCODE has been widely applied for calibration and sensitivity analysis of groundwater flow and transport models, and hydrological models. Some more example applications include:

- Calibration of nitrogen fate and transport modelling for example in the Florida onsite sewage nitrogen reduction strategies study (Hazen and Sawyer, 2009).
- Automated calibration of a stream solute transport model: implications for interpretation of biogeochemical parameters (Scott et al. 2003)
- Sensitivity analysis and calibration of a distributed physically based rainfall-runoff model (Foglia et al. 2009). In this example performance of the statistical measures produced by UCODE has been compared to standard measure used to evaluate the goodness of fit for hydrological models, such as the Nash-Sutcliffe criterion.
- Calibration of models for simulation of Nitrogen Losses in Conventional and Advanced Soil-Based Onsite Wastewater Treatment Systems under Current and Changing Climate Conditions (Morales et al., 2016)
- Sensitivity analysis and parameter estimation of a watershed scale model to evaluate the Watershed-Scale Impacts of Nitrogen from On-Site Wastewater Systems (Geza et al., 2010).
- UCODE has also been linked to another integrated hydrological model, MIKE-SHE, and for example used to calibrate and validate a national hydrological model for Denmark (Henriksen et al., 2003) For this model "Three model versions with different assumptions on input data and parameter values were required until the performance of the final, according to pre-defined accuracy criteria, model was evaluated as being satisfactory. The paper highlights the methodological issues related to establishment of performance criteria, parameterization and assessment of parameter values from field data, calibration and validation test schemes. Most of the parameter values were assessed directly from field data, while about 10 'free' parameters were subject to calibration using a combination of inverse steady-state groundwater modelling and manual trial-and-error dynamic groundwater/surface water modelling. Emphasizing the importance of tests against independent data, the validation schemes included combinations of split-sample tests (another period) and proxy-basin tests (another area)." (Henriksen et al., 2003).

Among the case studies run during FREEWAT project, implementing Calibration module, hereafter a short description of Selisoo case study in Estonia (run by partner University of Tartu, by colleagues Andres Marandi, Florian Zaun, Argo Jõeleht, Maile Polikarpus, Marko Kohv).

Selisoo bog in northeastern Estonia is a good example of area, where the direct impact of underground mining activities on groundwater regime and therefore also on wetland water balance can be studied by groundwater modelling. A 430 km$^2$ area, where a wetland is located within 3 km of an underground oil-shale mine was selected with the aim of the modelling to assess the effect of underground mining and its possible future extension to a neighboring wetland. The main task is to calculate if the peat layers and the Quaternary sediments (Q) below the bog are sufficient to keep wetland from draining when underground mine, which operates at the depth of 55 to 60 m from the ground surface, reaches under the wetland.

The model that was developed in this study is based on the groundwater flow model of Marandi et al. (2013) and uses mainly the same input data for discretization. It covers an area of ca. 430

km² (27 × 16 km) in the north-eastern part of Estonia, ca. 20 km south of Jõhvi, being situated between the Gulf of Finland and Lake Peipsi. It includes Selisoo and Ratva bogs as well as 95 % of the area of the Estonia oil-shale underground mine.

The boundaries of the model were selected according to the presence of groundwater-level monitoring wells, resulting in a grid of 270 columns and 161 rows (cell size 100 × 100 m). With respect to the local hydrostratigraphy, 10 layers were differentiated (Fig. 8.3). The first 3 layers (bog and Quaternary sediment) where created to simulate the bog sediments with rapidly decreasing hydraulic conductivities from top to bottom in contrast to the other Quaternary deposits. The model top is defined by a digital elevation model (DEM) and the areal distribution and thicknesses of the horizons were gathered from drilling data and geophysics (GPR for peat thickness) as well as results of previous studies of the oil-shale region and geological mapping (bedrock). Except for the first layer, which was defined as convertible, all other layers are simulated as confined aquifers in the Layer Property Flow Package (LPF).

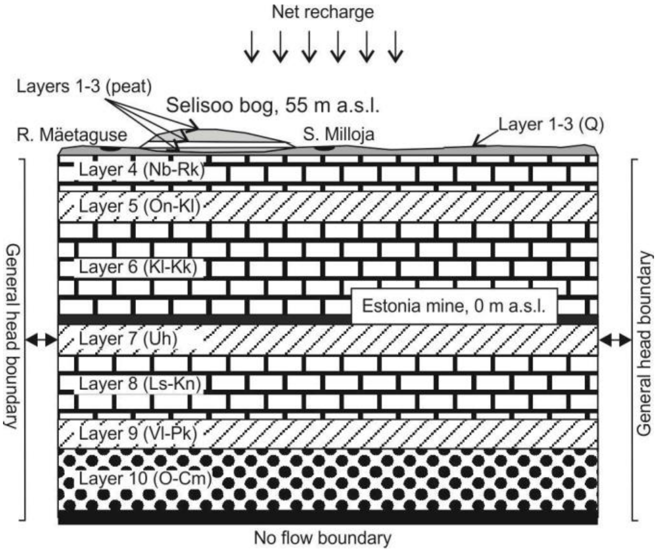
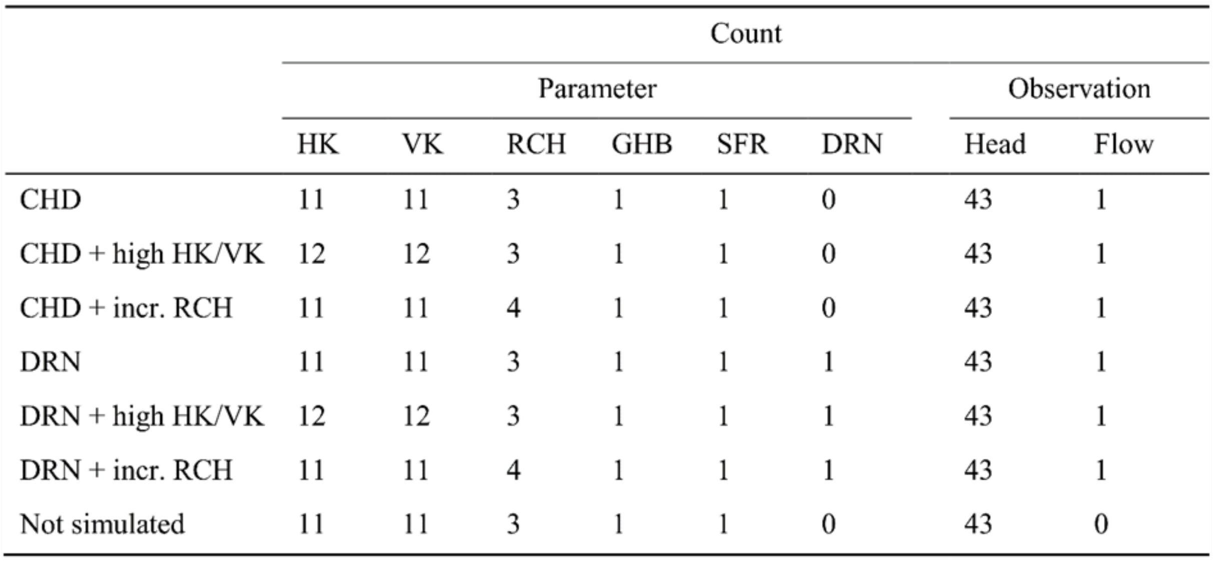


*Figure 8.3 - Conceptual cross-section of the groundwater model relating the model layers and geology in the area*

When modeling natural systems with incomplete data, such as in Selis0o case study, there are always several possible representations of it. (Hill & Tiedeman, 2007) highly recommend developing and discriminating between alternative models to solve problems concerning different equally plausible interpretations of incomplete system information or problems with model fit or optimal parameter values. In this study alternative models were developed in a deterministic way, comparing different approaches of boundary representations.  Due to the varying design of the alternative models, the number of parameters was changing in different models (Tab. 8.1).



| | Count | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Parameter | | | | | | Observation | |
| | HK | VK | RCH | GHB | SFR | DRN | Head | Flow |
| CHD | 11 | 11 | 3 | 1 | 1 | 0 | 43 | 1 |
| CHD + high HK/VK | 12 | 12 | 3 | 1 | 1 | 0 | 43 | 1 |
| CHD + incr. RCH | 11 | 11 | 4 | 1 | 1 | 0 | 43 | 1 |
| DRN | 11 | 11 | 3 | 1 | 1 | 1 | 43 | 1 |
| DRN + high HK/VK | 12 | 12 | 3 | 1 | 1 | 1 | 43 | 1 |
| DRN + incr. RCH | 11 | 11 | 4 | 1 | 1 | 1 | 43 | 1 |
| Not simulated | 11 | 11 | 3 | 1 | 1 | 0 | 43 | 0 |

*Table 8.1 - Number of parameters and observations for the seven different alternative models. HK – horizontal hydraulic conductivity, VK – vertical hydraulic conductivity, RCH – recharge rate, GHB – General Head Boundary conductance, SFR – streambed conductivity, DRN – drain conductance.*

Composite scaled sensitivities (CSS) and parameter correlation coefficients (PCC) were determined with starting parameter values before any parameter estimation was conducted, to select the most five influencing parameters, namely the ones named: HK_Par2, HK_Par4, VK_Par1, VK_Par3, RCH_Main, RCH_Mine and RCH_Bog.  Except for *Not simulated*, these five parameters were included in the estimation process for all models. For *Not simulated* only two recharge parameters were included in the regression. The parameter estimation results are given in Tab. 8.2.

|  | HK_Par2 | HK_Par4 | VK_Par1 | VK_Par3 | RCH_Main | RCH_Mine | RCH_Bog |
|---|---|---|---|---|---|---|---|
| Starting value | 4.00E+01 | 5.00E+00 | 1.00E−03 | 3.00E−04 | 1.60E−04 | 8.20E−04 | 1.00E−04 |
| CHD | **4.00E+02** | 8.15E−01 | 3.67E−03 | 3.13E−04 | **3.20E−04** | – | – |
| CHD + high HK/VK | 2.07E+02 | 5.18E−01 | **1.00E−02** | 1.51E−04 | **3.20E−04** | – | – |
| CHD + incr. RCH | 1.20E+02 | 7.28E−01 | **1.00E−02** | 1.17E−04 | – | **1.60E−03** | – |
| DRN | **4.00E+02** | 7.81E−01 | 3.66E−03 | 1.45E−04 | 2.87E−04 | – | – |
| DRN + incr. RCH | 2.82E+02 | **5.00E−01** | 9.51E−03 | 1.45E−04 | – | **1.60E−03** | – |
| Not simulated | – | – | – | – | 1.60E−05 | – | 1.02E−04 |

*Table 8.2 - Summary of parameter values before and after parameter estimation. All parameters are given in m/d. Bold values indicate that this parameter reached a constraint during the estimation process. The dash indicates a parameter that was not included in the parameter estimation for this model.*

After successful parameter runs, simulated values and residuals were evaluated. Furthermore, for comparison of alternative models, AIC, AICC, BIC and SSWR statistics of all six models were evaluated. From this comparison the conclusion is that the lowest values for all four evaluated statistics of overall model fit has the model *DRN + incr. RCH*, which therefore seems to give the most accurate representation of the system.

## 8.4 Application of solute transport in the unsaturated zone

Knowledge of water and solute movement in the variably saturated soil near the earth surface is essential to understand man's environmental impact: an accurate description of unsaturated soil water movement is essential to derive proper management conditions for vegetation growth and environmental protection in agricultural and natural systems.

Addressing this issue is of crucial importance to apply FREEWAT for facing problems in which surface source of pollution is present, due to intentional or accidental release of surface-applied and soil-incorporated chemicals into the environment. Typical examples are listed hereafter.

- Characterization of contaminated sites, to evaluate leaching of contaminants from soil to ground water, e.g. benzene, toluene, ethylbenzene and xylenes (BTEX), or radionuclides emanating from nuclear waste disposal facilities (Healy, 2007). A practical example of modeling application is the one reported in Kresic (2006). A granular solid contaminant was deposited at the land surface for 15 years at a constant annual loading rate of 20.9 mg/m2. The contaminant is known to be recalcitrant (not readily biodegradable) and so a numeric model for both the vadose zone and the saturated zone below the source area was developed, to predict contaminant dissolved concentrations as it leaches from the land surface, for the first 15 years of active contaminant loading, followed by additional 15 years after the loading was discontinued. So, the primary objective of the unsaturated-saturated groundwater model is to demonstrate how the contaminant residing on the ground surface can leach to the saturated zone through the vadose zone, and subsequently migrate in the saturated zone off the immediate aquifer portion underlying the source area. The most interesting achievement of this modeling study was the effect of a clay lens present in the subsurface on contaminant fate and transport: due to low hydraulic conductivity and contaminant sorption, the clay lens acts as a long-term storage, and a secondary source of dissolved contaminant in the saturated zone.
- Environmental impact of a landfill on groundwater. An example of application is reported in Anwar and Thien (2015). This modeling study investigated the potential leachate of heavy metals from the landfill site of Rottnest Island in Western Australia. This landfill was unlined, and this can easily cause leachate migration from the landfill during rain events and can contaminate underlying groundwater. In the meantime, the main freshwater resources at Rottnest Island are the two shallow unconfined aquifers known as Wadjemup mounds. Boreholes were set up around the landfill site for groundwater monitoring. The groundwater sampling from the boreholes showed that there were heavy metals concentrations exceeding the guidelines given by ANZECC for fresh and marine water quality, especially for copper and zinc concentration. For this reason, the contaminant migration of these two heavy metals were modelled for different solute transport parameters including initial concentration, soil adsorption coefficient, longitudinal dispersion, saturated hydraulic conductivity, initial soil water content and water diffusion coefficient. The results revealed that soil adsorption parameter is the most dominating factors for leachate transport in landfill site. The initial concentration also shows significant effect on mass transport in soil. The longitudinal dispersion was found affecting moderately on leachate migration but water diffusion coefficient has no effect. The initial soil moisture

and saturated hydraulic conductivity shows minimum effect on leachate transport in soil.

- Studies involving prescription and recommendation given by EU Nitrate Directive, in rural/agricultural context, to respect mandatory standards, such as the maximum permissible nitrate concentration in groundwater (50 mg/L). Examples are reported in Tafteh and Sepaskhah (2012) or Perego et al. (2011). The latter had the objective of quantifying the nitrate leaching and its relation with ordinary management in 6 fields with contrasting pedoclimatic conditions under grain and silage maize over a period of 2–5 years in Po Valley (Italy). The collected data sets were also suitable to be used as input in a modelling application, being representative of the studied area, to estimate the drainage flux.

A synthetic test applying UZT package is here presented, to show the successful application of FREEWAT as professional tool to address contaminated sites modeling.
The aim is to estimate the effect of a prescribed source of contaminant (or tracer, in general), released at the ground surface. Due to rainwater infiltration, this contaminant can infiltrate through the vadose zone, eventually reaching the aquifer, where it will spread according to the groundwater flow. In particular, such a situation refers to a decommissioning area, where a soil contamination is detected. The complete removal of this contamination source took 90 days.
The local Environmental Protection Agency wanted to be sure that the contaminant plume (due to the pollutant leaching) did not spread to an adjacent area, which is going to host a new building block, whose construction will commence within 1 year. A picture showing the test area is reported in Fig. 8.4.

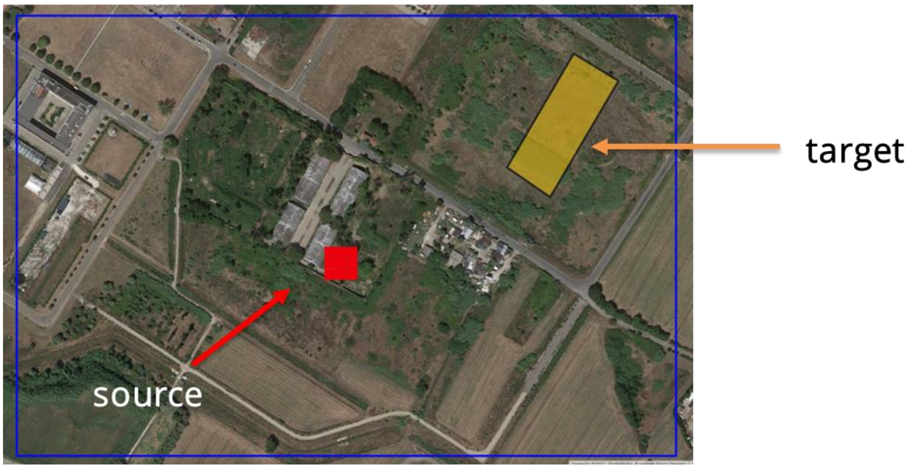


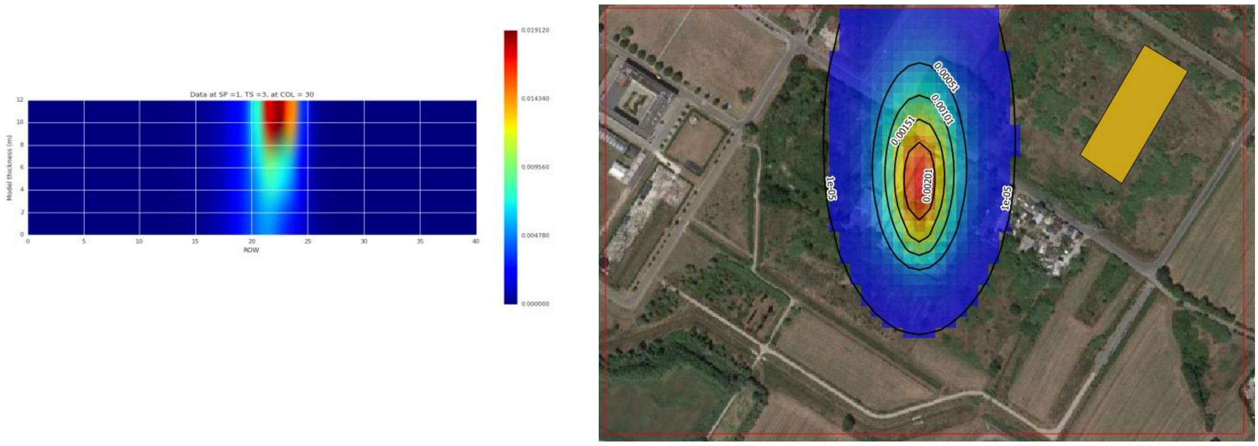
*Figure 8.4 - The decommissioned are where the contamination source was detected.*

A solute transport numerical model is decided to get a technical sounding decision about the need of designing and setting in action a remediation plan or, conversely, to claim that natural attenuation will allow avoid a specific remediation activity.
The model represents the hydrologic structure of the site as a 3-layer system. The middle layer serves to better represent a clay silty-sand lens of low-conductivity, which could prevent the large spreading of the contaminant. The 360-day time range of simulation is divided in two stress periods: the first one (90 day) with contamination active, and the second one (270 day)

without contamination. The other only stress on the system is the aquifer recharge due to precipitation, modeled by UZF package (so that the flux in unsaturated zone can be calculated as well).

In FREEWAT, the MOFLOW model including UZF package is easily implemented; after the solution of the hydrological problem, the latter is coupled with a transport problem, including only one species, having a (normalized) concentration in water of 1 Kg/m$^3$.

Model results capture the slowing effect of the silty-clay lens (well evident if a cross section plot is made, using some of the FREEWAT visualization functionalities, Fig. 8.5) and give the answer to the initial question: in layer 1 (the most affected by contamination) concentration at the end of Stress Period 2 (time Step 9) has a maximum value of around 0.0025 mg/ml (0.25% of the initial concentration at ground surface). This gives evidence of the ineffective presence of pollution, with respect to the position of our target (Fig. 8.5). Therefore, at within the 360-day time range, remediation of the target site is not required.



(a) (b)

*Figure 8.5 - A sample of results obtained by FREEWAT when running the synthetic test on solute transport in vadose zone. (a) Cross section of contamination spread at the end of stress period 1 (90 days). (b) Raster representation of the contamination plume at the end of simulation (360 days) in layer 1 (top layer), and its positioning w.r.t. the target zone.*

## 8.5 Application of akvaGIS

The Hydrochemical Analysis Tools included in AkvaGIS can be used in all kind of projects. Scientific community, public administration (e.g. Geological Surveys) and also the private sector (e.g. water management, water supply, mining control, etc.) can get benefit from the easy way that the tools create and customize reports and quickly analyse plots and diagrams.

There are several environmental assessments where is of crucial importance to apply FREEWAT (specifically AkvaGIS tools) to store, share and manage data related to quality of water. Research works, such as Puig et al., 2013 or Scheiber et al., 2015; 2016, in which hydrochemical analysis of water quality were developed, could take advantage of these tools. For instance, a potential user can be the current project called Gro for GooD: Groundwater Risk Management for Growth and Development (https://upgro.org/consortium/gro-for-good/), in which different partners and stakeholders have to share and use huge amount of data in the same format from the beginning of the project.

FREEWAT project partner UTARTU (Tartu, Estonia), with the support of IDAEA-CISIC (Barcelona, Spain), applied akvaGIS tool to create a database of Estonian groundwater body monitoring points. This database includes 277 groundwater monitoring wells (Fig. 8.6) from all groundwater bodies with daily groundwater level measurements and chemical analyses, which were collected within year 2015. This database was used in the local FREEWAT training courses (during the H2020 FREEWAT project) as a database example, so that anyone could download and use it. Participants of the training courses found AkvaGIS to be a useful tool for their work.

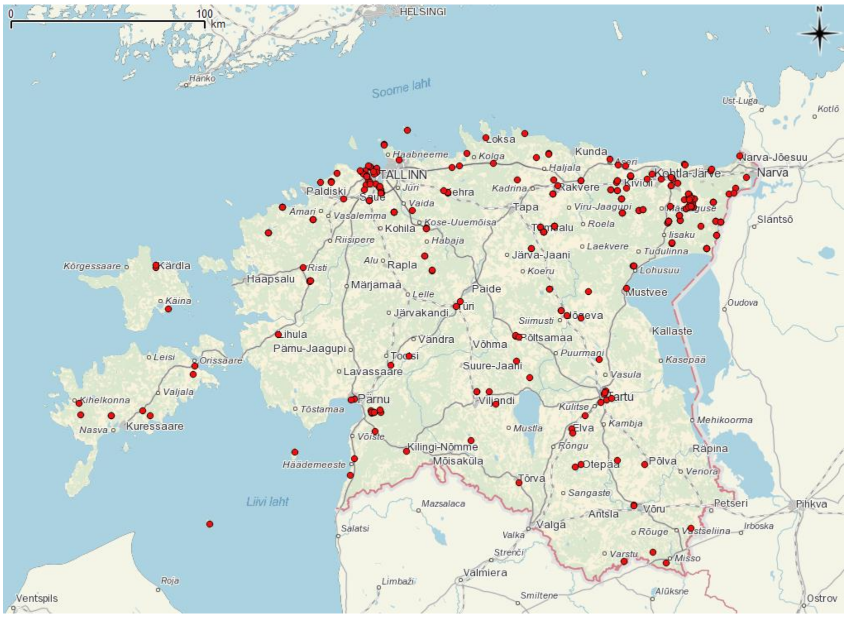


*Figure 8.6 - Estonian application of the AkvaGIS tools. Red dots are points with hydrochemical information.*

In the framework of the project: “Groundwater contribution to greenhouse gas (GHGs) emissions from rivers” (Walloon Region, Belgium), AkvaGIS was applied to understand the hydrochemical groundwater characteristics performing hydrochemical diagrams (i.e. Piper plot) and spatial analysis (i.e. STIFF diagram maps), Figs. 8.7 and 8.8. The AkvaGIS database filled will be used to store, manage and analyze future field campaigns in the area. The physical and chemical data stored from 64 groundwater samples, were uploaded to the AkvaGIS database after the field campaign.

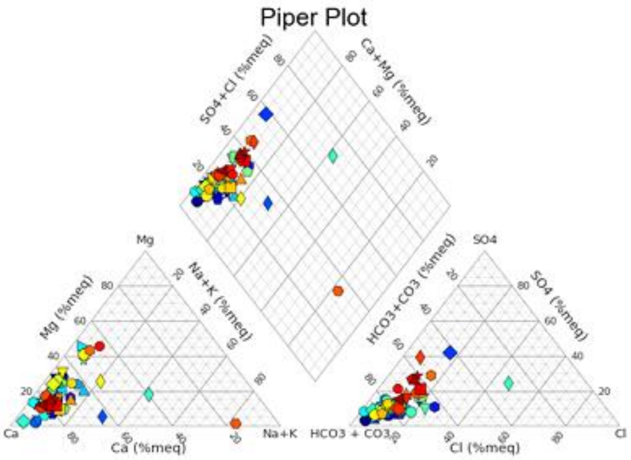


*Figure 8.7 - Piper plot of the sampling points from the 2016 campaign in the Walloon Region. Belgium.*

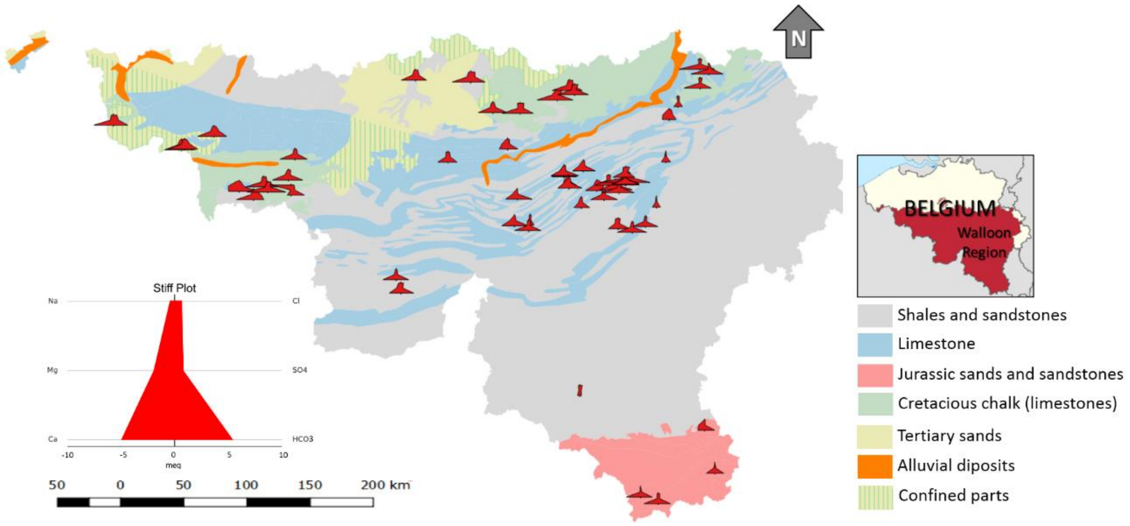


*Figure 8.8 - Spatial distribution of the Stiff diagram of the Walloon region aquifers for the campaign of 2016 in the Walloon Region. Belgium.*

## 8.6 Application of OAT module

OAT as a part of FREEWAT will be demonstrated through the case study of SUPSI-IST. Fig. 8.1 of the Lake section (above) shows the location of monitoring stations administered by SUPSI-IST as well as the state administered monitoring stations within the Lake lugano watershed. Temperature, precipitation, streamflow, stream height, humidity, atmospheric pressure, and lake level, among others, are monitored through istSOS. For the Lake Lugano case study, several of these are used to generate inputs for the model. Additionally, other sensor data can be used for model calibration at a later stage. Precipitation data from the sensors was imported from istSOS and, in conjunction with land use data, used to calculate the recharge applied to the model cells of the aquifers. Temperature data was used with a newly added process to calculate evaporation through the Hargreaves-Samani equation (Allen et. al. 1998), and was used for evaporation from the lakes. Streamflow gages were used to estimate the inflow into streams at the model boundary for those streams that originate outside of the model area. This was done largely based on the relative surface area of gaged and ungaged sites. All of the above time series were additionally automatically resampled to conform to the chosen temporal discretisation of the model; monthly stress periods and daily time steps. The resampled time series were used as model inputs for the various MODFLOW packages. The case study will also use OAT for the automatic conversion of sensor time series data to head observations package input during the model calibration phase, using ground water elevations recorded by SUPSI-IST.

# References

Allen R.G., Pereira L.S., Raes D. and Smith M. (1998). Crop evapotranspiration-Guidelines for computing crop water requirements-FAO Irrigation and drainage paper 56. FAO, Rome, (56), 300.

Anderman E.R., Hill M.C. and Poeter E.P., 1996, Two-dimensional advective transport in groundwater flow parameter estimation: Ground Water, 34(6):1001-1009.

Anwar F. and Thien L. (2015), Investigating Leachate Transport at Landfill Site Using HYDRUS-1D," International Journal of Environmental Science and Development, 6 (10), 741-745.

Barlebo H.C., Hill M.C., Rosbjerg D. and Jensen K.H. (1998). Concentration data and dimensionality in groundwater models: Evaluation using inverse modeling: Nordic Hydrology, v. 29, p. 149-178.

Barlebo H.C., Hill M.C. and Rosbjerg Dan (2004). Investigating the Macrodispersion Experiment (MADE) site in Columbus, Mississippi, using a three-dimensional inverse flow and transport model: Water Resources Research, Vol. 40, No. 4, W0421110.1029/2002WR001935.

Barth G.R. and Hill M.C. (2005a). Numerical methods for improving sensitivity analysis and parameter estimation of virus transport simulated using sorptive-reactive processes: Journal of Contaminant Hydrology, v. 76, p. 251-277.

Barth G.R. and Hill M.C. (2005b). Parameter and observation importance in modeling virus transport in saturated systems investigations in a homogenous system: Journal of Contaminant Hydrology, v. 80, p. 107-129.

Bedekar V., Morway E.D., Langevin C.D., Tonkin M. (2016), MT3D-USGS version 1: A U.S. Geological Survey release of MT3DMS updated with new and expanded transport capabilities for use with MODFLOW: U.S. Geological Survey Techniques and Methods 6-A53, 69 p.

Cannata M. and Antonovic M. (2010). istSOS: investigation of the sensor observation service. In WebMGS 1st international workshop on pervasive web mapping, geoprocessing and services, Como, Italy (pp. 26-27).

Cannata, M. and Antonovic M. (2015). Sensor Observations Service for Environmentally Optimizing Irrigation: istSOS within the ENORASIS Example. International Journal of Geoinformatics, 11(3).

Cannata M., Antonovic M., Molinari M. and Pozzoni M. (2015). istSOS, a new sensor observation management system: software architecture and a real-case application for flood protection. Geomatics, Natural Hazards and Risk, 6(8), 635-650.

Cheng, X. and Anderson M. P. (1993). Numerical Simulation of Ground-Water Interaction with Lakes Allowing for Fluctuating Lake Levels. Ground Water, 31(6), 929-933.
Cocca P. A., Doherty J. and Kittle J. L. (2003). Hydrologic Calibration Strategies for the HSPF Watershed Model: Identifying Effective Objective Functions for Use with the Parameter Estimation (PEST) Program. In Proceedings World Water & Environmental Resources Congress 2003.

Council, G. W. (1997). Simulating lake-groundwater interaction with MODFLOW , in Proceedings of the 1997 Georgia Water Resources Conference, ed. by K. J. Hatcher, pp. 6 p., The University of Georgia.

D'Agnese F.A., Faunt C.C., Hill M.C. and Turner A.K. (1999). Death Valley regional ground-water flow model calibration using optimal parameter estimation methods and geoscientific information systems: Invited paper for a Special Section on Model Calibration and Reliability Evaluation for Ground-Water Systems, eds. A. Leijnse and M.C. Hill, Advances in Water Resources, vol. 22, no. 8, p. 777-790.

Doherty J. (2010). PEST, Model-independent parameter estimation—User manual (5th ed., with slight additions) and addendum: Brisbane, Australia, Watermark Numerical Computing. http://www.sspa.com/PEST/index.html

Doherty J. (2004), PEST: Model-independent parameter estimation, User manual: 5th ed. Watermark Numer. Comput., Brisbane, Queensl., Australia.

Durelle S.T., Gooseff M.N., Bencala K.E. and Runkel R.L. (2003). Automated calibration of a stream solute transport model: implications for interpretation of biogeochemical parameters, Journal of the North American Benthological Society 2003 22:4, 492-510.

El-Zehairy, A.A.M.E. (2014) Assessment of lake - groundwater interactions : Turawa case, Poland. Enschede, University of Twente Faculty of Geo-Information and Earth Observation (ITC), 2014.

European Community. (2000). Directive 2000/60/EC of the European Parliament and of the Council of 23 October 2000 establishing a framework for Community action in the field of water policy. Official Journal of the European Parliament, L327(September 1996), 1–82. http://doi.org/10.1039/ap9842100196

European Commission (2009). Common implementation strategy for the water framework directive (WFD) (2000/60/EC). Guidance document nº 22. Update guidance on implementing the geographical information system (GIS) elements of the EU Water Policy. Technical report-2009-028. ISBN 978-92-79-11373-4.

Faunt C.C. Ed. (2009), Groundwater Availability of the Central Valley Aquifer, California: U.S. Geological Survey Professional Paper 1766.

Fenske J. P., Leake S. A. and Prudic D. E. (1996). Documentation of a Computer Program (RES1) to Simulate Leakage from Reservoirs Using the Modular Finite-Difference Ground-Water Flow Model (MODFLOW). U.S. Geological Survey, (Open-File Report 96-364), 51. Retrieved from http://water.usgs.gov/software/MODFLOW/code/doc/ofr96364.pdf

Fienen M.N., D'Oria M., Doherty J.E. and Hunt R.J. (2013). Approaches in highly parameterized inversion: bgaPEST, a Bayesian geostatistical approach implementation with PEST—Documentation and instructions: U.S. Geological Survey Techniques and Methods, book 7, section C9 , 86 p. Documentation available at http://pubs.usgs.gov/tm/07/c09. Program available at: https://github.com/mnfienen/bgaPEST.

Foglia L., Mehl S.W., Hill M.C., Perona P., Burlando P. (2007). Testing alternative ground water models using cross validation and other methods: Ground Water, 45(5): 627-641, doi: 10.1111/j.1745-6584.2007.00341.x.

Foglia L., M.C. Hill, S. W. Mehl, and P. Burlando (2009). Sensitivity analysis, calibration, and testing of a distributed hydrological model using error-based weighting and one objective function, Water Resour. Res., 45, W06427, doi:10.1029/2008WR007255

Foster, S.S.D. (2001). The interdependence of groundwater and urbanization in rapidly developing cities. Urban Water, 3(3), 185–192.

Geza Mengistu, M. CE; Kyle E. Murray; and John E. McCray, M.ASCE (2010). Watershed-Scale Impacts of Nitrogen from On-Site Wastewater Systems: Parameter Sensitivity and Model Calibration, Journal of Environmental Engineering, Volume 136 Issue 9, http://ascelibrary.org/doi/abs/10.1061/(ASCE)EE.1943-7870.0000232#sthash.5naglZt1.dpuf

Gooseff, M. N., Anderson J.K., Wondzell S.M., LaNier J., Haggerty R. (2005). A modelling study of hyporheic exchange pattern and the sequence, size, and spacing of stream bedforms in mountain stream networks, Oregon, USA . Hydrological Processes. 2915-2929, DOI: 10.1002/hyp.5790. (Pub No: 3863)

Hanson R.T., Schmid W., Faunt C.C., and Lockwood B. (2010), Simulation and Analysis of Conjunctive Use with MODFLOW's Farm Process, Ground Water 48(5):674-89

Hanson R.T., Boyce S.E., Schmid W., Hughes J.D., Mehl S.M., Leake S.A., Maddock T., Niswonger R.G. (2014). One-Water Hydrologic Flow Model (MODFLOW-OWHM). U.S. Geological Survey, Reston, Virginia.

Harbaugh A.W. (2005). MODFLOW-2005, The U.S. Geological Survey Modular Ground-Water Model—the Ground-Water Flow Process. U.S. Geological Survey, Reston, Virginia.

Hazen and Sawyer, 2009, Florida Onsite Sewage Nitrogen Reduction Strategies Study TASK D.2 FINAL REPORT Literature Review of Nitrogen Fate and Transport Modeling, Florida Department of Health.

Healy R.W. (2007). Simulating Water, Solute, and Heat Transport in the Subsurface with the VS2DI Software Package, Vadose Zone Journal, 7 (2), 632-639.

Henriksen Hans Jørgen, Lars Troldborg, Per Nyegaard, Torben Obel Sonnenborg, Jens Christian Refsgaard, Bjarne Madsen (2003). Methodology for construction, calibration and validation of a national hydrological model for Denmark, Journal of Hydrology, Volume 280, Issues 1–4, Pages 52-71, ISSN 0022-1694, http://dx.doi.org/10.1016/S0022-1694(03)00186-0

Hill M.C. (1998). Methods and guidelines for effective model calibration: U.S Geological Survey Water- Resources Investigations Report 98-4005, 90p. http://pubs.water.usgs.gov/wri984005/

Hill M.C. (2006). The practical use of simplicity in developing ground-water models: Ground Water, 44(6):775-781.

Hill M.C. and Tiedeman C.R. (2007). Effective groundwater model calibration, with analysis of sensitivities, predictions, and uncertainty: Wiley and Sons, New York, New York, 455 p.

Hunter J.D. (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95, DOI:10.1109/MCSE.2007.55

LIFE NITRATES – Repercussions of agricultural practices on the nitrate pollution of inland waters (LIFE+10 ENV/ES/478) (2016), http://www.life-nitratos.eu/index.php/en/ [last access: June 2016]

Kim Y. J., Kim H. D. and Jeon J. H. (2014). Characteristics of water budget components in paddy rice field under the Asian monsoon climate: Application of hspf-paddy model. Water, 6(7), 2041-2055.

Kresic N. (2006). Hydrogeology and Groundwater Modeling, Second Edition, CRC Press.

Krohelski J. T., Lin Y. F., Rose W. J. and Hunt R. J. (2002). Simulation of Fish, Mud, and Crystal Lakes and the shallow ground-water system, Dane County, Wisconsin (No. 2002-4014). US Geological Survey.

Matott L. S. (2010). OSTRICH: An Optimization Software Tool. Accessed 12/27/2010 at http://www.civil.uwaterloo.ca/lsmatott/Ostrich/OstrichMain.html

McKinney W. (2010). Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56

Mehl S.W. and Hill M.C. (2001). A comparison of solute-transport solution techniques and their effect on sensitivity analysis and inverse modeling results: Ground Water, 39(2): 300-307.

Mendizabal I., Stuyfzand P. J. (2009). Guidelines for interpreting hydrochemical patterns in data from public supply well fields and their value for natural background groundwater quality determination. Journal of Hydrology, 379(1-2), 151–163. doi:10.1016/j.jhydrol.2009.10.001.

Merritt M. L. and Konikow L. F. (2000). Documentation of a computer program to simulate lake-aquifer interation using the MODFLOW Ground-Water Flow Model and the MOC3D Solute-Transport Model. U.S. Geological Survey Water-Resources Investigations Report 00-4167, 146.

Morales I., Cooper J., Amador J.A. and Boving T.B. (2016). Modeling Nitrogen Losses in Conventional and Advanced Soil-Based Onsite Wastewater Treatment Systems under Current and Changing Climate Conditions. PLoS ONE, 11(6), e0158292. http://doi.org/10.1371/journal.pone.0158292

Morway E.D., Niswonger R.G., Langevin C.D., Bailey R.T., Healy R.W. (2013). Modeling Variably Saturated Subsurface Solute Transport with MODFLOW-UZF and MT3DMS, GROUNDWATER, Vol. 51, no. 2.

Niswonger, R.G., Prudic D.E., and Regan R.S. (2006). Documentation of the Unsaturated-Zone Flow (UZF1) Package for modeling unsaturated flow between the land surface and the water table with MODFLOW-2005. U. S. Geological Survey Techniques and Methods 6-A19, 62 p.

Niswonger R.G. and Prudic D.E. (2010). Documentation of the Streamflow-Routing (SFR2) Package to Include Unsaturated Flow Beneath Streams—A Modification to SFR1: U.S. Geological Survey Techniques and Methods 6-A13. US Geological Survey Techniques and Methods 6, (April 2010), 59. Retrieved from http://pubs.er.usgs.gov/publication/tm6A13

Park H.S., San Juan C., (2000). A method for assessing leaching potential for petroleum hydrocarbons release sites: multiphase and multi-substance equilibrium partitioning, Soil and Sediment Contamination, 9(6), 611-632, 2000.

Perego A., Basile A., Bonfante A., De Mascellis R., Terribile F., Brennac S., Acutis M. (2011). Nitrate leaching under maize cropping systems in Po Valley (Italy), Agriculture, Ecosystems and Environment 147, 57– 65.

Poeter E.P. and Hill M.C. (1997). Inverse models: a necessary next step in groundwater modeling, Ground Water: v. 35(2), p. 250-260

Poeter E.P., Hill M.C., Lu D., Tiedeman C.R. and Mehl S. (2014). UCODE_2014, with new capabilities to define parameters unique to predictions, calculate weights using simulated values, estimate parameters with SVD, evaluate uncertainty with MCMC, and More: Integrated Groundwater Modeling Center Report Number: GWMI 2014-02.

Puig R., Folch A., Menció A., Soler A., Mas-Pla J. (2013). Multi-isotopic study (15N, 34S, 18O, 13C) to identify processes affecting nitrate and sulfate in response to local and regional groundwater mixing in a large-scale flow system, Applied Geochemistry, Volume 32, Pages 129-141, ISSN 0883-2927. doi.org/10.1016/j.apgeochem.2012.10.014.

QGIS Development Team, (2016). QGIS Geographic Information System. Open Source Geospatial Foundation Project. http://qgis.osgeo.org

Rossetto R., Borsi I. and Foglia L. (2015). FREEWAT: FREE and open source software tools for WATer resource management. Rendiconti Online Della Società Geologica Italiana, 35, 252–255. http://doi.org/10.3301/ROL.2015.113

Scheiber L., Ayora C., Vázquez-Suñé E., Cendón D. I., Soler A., Custodio E. and Baquero J. C. (2015). Recent and old groundwater in the Niebla-Posadas regional aquifer (southern Spain): Implications for its management. Journal of Hydrology, 523, 624-635.

Scheiber L., Ayora C., Vázquez-Suñé E., Cendón D. I., Soler A., & Baquero J. C. (2016). Origin of high ammonium, arsenic and boron concentrations in the proximity of a mine: Natural vs. anthropogenic processes. Science of the Total Environment, 541, 655-666.

Schmid W., Hanson R.T., Maddock T., Leake S.A. (2006). User Guide for the Farm Process (FMP1) for the U.S. Geological Survey's Modular Three-Dimensional Finite-Difference Ground-Water Flow Model, MODFLOW-2000. U.S. Geological Survey, Reston, Virginia.

Schmid W. and Hanson R.T. (2009). The Farm Process Version 2 (FMP2) for MODFLOW 2005 - Modifications and Upgrades to FMP1. U.S. Geological Survey, Reston, Virginia.

Searcy JK (1959). Flow-duration curves, Manual of hydrology, pt. 2, Low-flow techniques. U.S. Geological Survey Water-Supply Paper 1542–A, 32 p.

Skahill, B. E., Downer, C. W., & Baggett, J. S. (2012). A Practical Guide to Calibration of a GSSHA Hydrologic Model Using ERDC Automated Model Calibration Software-Efficient Local Search (No. ERDC/CHL-TR-12-3). ENGINEER RESEARCH AND DEVELOPMENT CENTER VICKSBURG MS COASTAL AND HYDRAULICS LAB.

Tafteh A., Sepaskhah R. A. (2012). Application of HYDRUS-1D model for simulating water and nitrate leaching from continuous and alternate furrow irrigated rapeseed and maize fields, Agricultural Water Management 113, 19-29.

Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30, DOI:10.1109/MCSE.2011.37

Vázquez-Suñé E., Casamitjana A., Sánchez-Vila X., Melcion C., Alcolea A., Sanz E. (2005). Hidrogeología de Badalona. Àmbit de Medi Ambient i Sostenibilitat .Ajuntament de Badalona. Badalona, Spain, 83 pp.

Velasco V., Tubau I., Vázquez-Sunñé E., Gogu R., Gaitanaru D., Alcaraz M., Sánchez-Vila X. (2014). GIS-based hydrogeochemical analysis tools (QUIMET). Computers & Geosciences, 70, 164-180.

Virdi M. L., Lee T. M. (2009). Simulating Lake-Groundwater Interactions During Decadal Climate Cycles: Accounting For Variable Lake Area In The Watershed. AGU Fall Meeting Abstracts

Virdi M. L., Lee T. M., Swancar A. and Niswonger R. G. (2013), Simulating the Effect of Climate Extremes on Groundwater Flow Through a Lakebed. Groundwater, 51: 203–218.

Walker J. F., Hunt R. J. and Doherty J. (2007). Effective Use of Time-Series Data to Calibrate a Coupled Ground Water/Surface Water Model in a Small Headwater Watershed, Northern Wisconsin. In AGU Fall Meeting Abstracts (Vol. 1, p. 0700).

Welter D.E., Doherty J.E., Hunt R.J., Muffels C.T., Tonkin M.J. and Schreüder W.A. (2012). Approaches in highly parameterized inversion: PEST++, a Parameter ESTimation code optimized for large environmental models. U.S. Geological Survey Techniques and Methods, book 7, section C5, 47 p. http://wi.water.usgs.gov/models/pestplusplus/

Westenbroek S. M., Doherty J. E., Walker J. F., Kelson V. A., Hunt R. J. and Cera T. B. (2012). Approaches in highly parameterized inversion: TSPROC, a general time-series processor to assist in model calibration and result summarization. U.S. Geological Survey Techniques and Methods, (7-C7), 112.

Yager R.M. (1998). Detecting influential observations in nonlinear regression modeling of groundwater flow. Water Resources Research 34 (7), 1623–1633.

Yager, R.M. (2004). Effects of model sensitivity and nonlinearity on nonlinear regression of ground water flow, Ground Water, v. 42, no. 3, p. 390-400.http://www3.interscience.wiley.com/cgi-bin/fulltext/118752546/PDFSTART

Zheng C. and Wang P.P. (1999). MT3DMS, A modular three-dimensional multi-species transport model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater systems; documentation and user's guide, U.S. Army Engineer Research and Development Center Contract Report SERDP-99-1, Vicksburg, MS, 202 p.

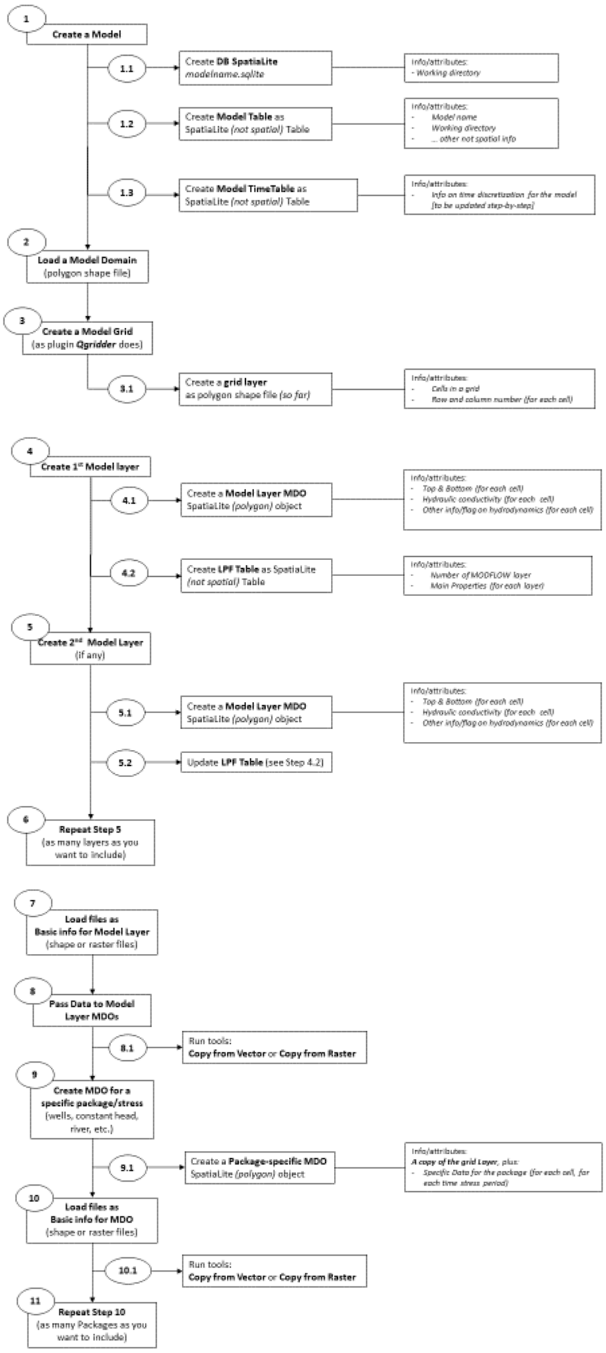
# Document History

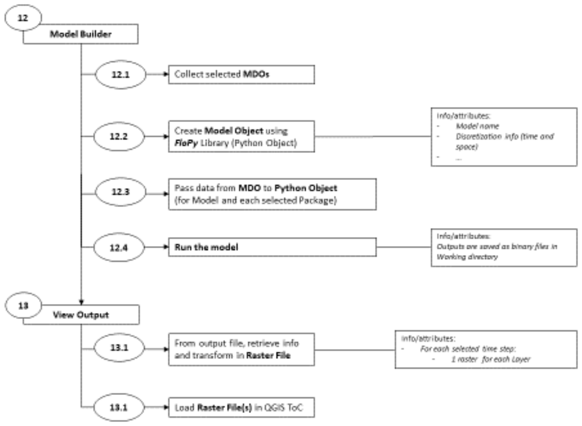
| | |
|---|---|
| 30.09.2017 | Release of v1.0 |

# Appendix A Programmer documentation for Groundwater Modelling

The main part of FREEWAT plugin is conceived to serve as pre- and post-processor for MODFLOW-2005 code.

The modelling workflow to execute a groundwater flow model in FREEWAT is schematize in the following diagram.


1
Create a Model
1.1
Create DB SpatiaLite
modelname.sqlite
Info/attributes:
- Working directory
1.2
Create Model Table as
SpatiaLite (not spatial) Table
Info/attributes:
- Model name
- Working directory
- ... other not spatial info
1.3
Create Model TimeTable as
SpatiaLite (not spatial) Table
Info/attributes:
- Info on time discretization for the model
(to be updated step-by-step)
2
Load a Model Domain
(polygon shape file)
3
Create a Model Grid
(as plugin Qgridder does)
3.1
Create a grid layer
as polygon shape file (so far)
Info/attributes:
- Cells in a grid
- Row and column number (for each cell)
4
Create 1st Model layer
4.1
Create a Model Layer MDO
SpatiaLite (polygon) object
Info/attributes:
- Top & Bottom (for each cell)
- Hydraulic conductivity (for each cell)
- Other info/flag on hydrodynamics (for each cell)
4.2
Create LPF Table as SpatiaLite
(not spatial) Table
Info/attributes:
- Number of MODFLOW layer
- Main Properties (for each layer)
5
Create 2nd Model Layer
(if any)
5.1
Create a Model Layer MDO
SpatiaLite (polygon) object
Info/attributes:
- Top & Bottom (for each cell)
- Hydraulic conductivity (for each cell)
- Other info/flag on hydrodynamics (for each cell)
5.2
Update LPF Table (see Step 4.2)
6
Repeat Step 5
(as many layers as you
want to include)
7
Load files as
Basic info for Model Layer
(shape or raster files)
8
Pass Data to Model
Layer MDOs
8.1
Run tools:
Copy from Vector or Copy from Raster
9
Create MDO for a
specific package/stress
(wells, constant head,
river, etc.)
9.1
Create a Package-specific MDO
SpatiaLite (polygon) object
Info/attributes:
A copy of the grid Layer, plus:
- Specific Data for the package (for each cell, for
each time stress period)
10
Load files as
Basic info for MDO
(shape or raster files)
10.1
Run tools:
Copy from Vector or Copy from Raster
11
Repeat Step 10
(as many Packages as you
want to include)

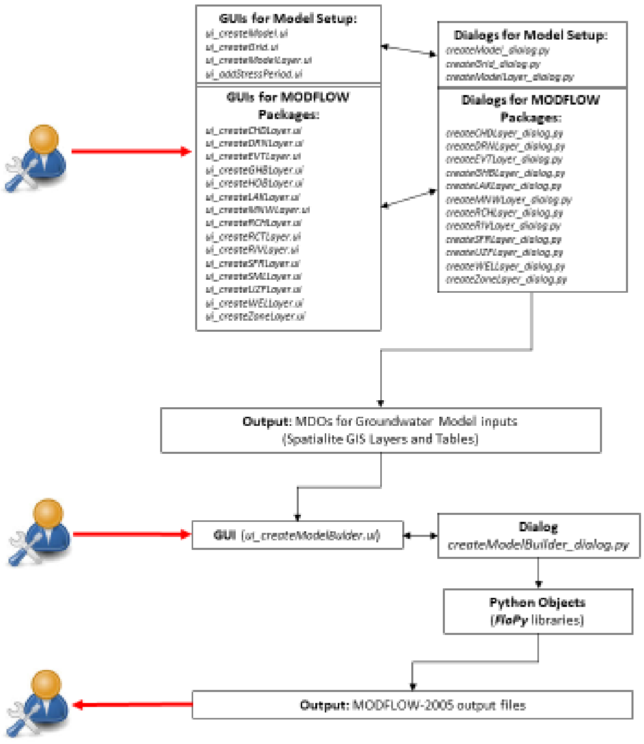

12
Model Builder
12.1
Collect selected MDOs
12.2
Create Model Object using FloPy Library (Python Object)
Info/attributes:
- Model name
- Discretization info (time and space)
- ...
12.3
Pass data from MDO to Python Object (for Model and each selected Package)
12.4
Run the model
Info/attributes:
Outputs are saved as binary files in Working directory
13
View Output
13.1
From output file, retrieve info and transform in Raster File
Info/attributes:
- For each selected time step:
- 1 raster for each Layer
13.1
Load Raster File(s) in QGIS ToC

The Python code belonging to this part of FREEWAT source code can be classified in three main groups:

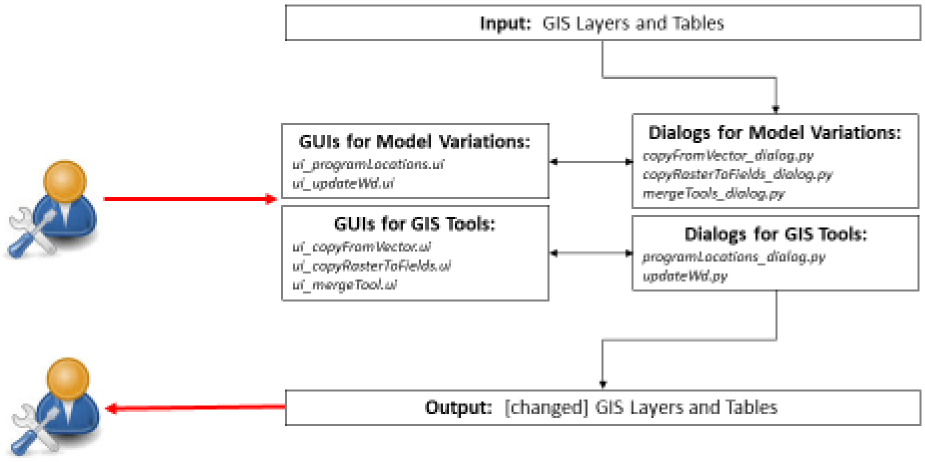
1. Code for defining and executing groundwater modelling
2. Code for executing model-related tools (to adjust or modify model settings)
3. Code for post-processing

For each of these branches, the link among different Python files is summarized in the following diagrams.

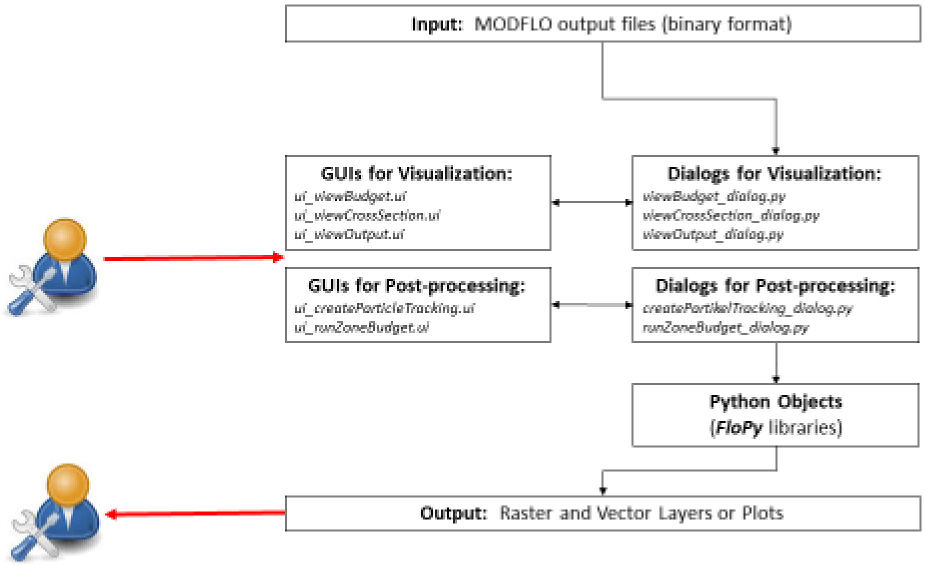
*Groundwater model definition and run:*



*Tools to change model settings or GIS-based tools:*



*Post-processing tools, including reading binary out files and post-processor codes (MODPATH and ZONEBUDGET):*


Input: MODFLO output files (binary format)
GUIs for Visualization:
ui_viewBudget.ui
ui_viewCrossSection.ui
ui_viewOutput.ui
Dialogs for Visualization:
viewBudget_dialog.py
viewCrossSection_dialog.py
viewOutput_dialog.py
GUIs for Post-processing:
ui_createParticleTracking.ui
ui_runZoneBudget.ui
Dialogs for Post-processing:
createPartikelTracking_dialog.py
runZoneBudget_dialog.py
Python Objects
(FloPy libraries)
Output: Raster and Vector Layers or Plots

# Appendix B Programmer documentation for Lake Package

*class class CreateLAKLayerDialog(QDialog, FORM_CLASS)*oatlib.sensor.Sensor

Initialize the interface

**Parameters:**

- **QDialog** (*obj*) – the Qgis dialog object
- **FORM_CLASS** (*str*) – user interface object

---------------------------------------------------------------------------------------------------

manageGui ()oatlib.sensor.Sensor.ts_zeros

create the GUI

---------------------------------------------------------------------------------------------------

check_if_exists ()oatlib.sensor.Sensor.ts_zeros

Check if a lak_layer_group is loaded in qgis

---------------------------------------------------------------------------------------------------

load_data_from_db ()oatlib.sensor.Sensor.ts_zeros

load the data from sqlite to gui

---------------------------------------------------------------------------------------------------

add_new_lake ()oatlib.sensor.Sensor.ts_zeros

Read data from spinBox and create a new lake

---------------------------------------------------------------------------------------------------

read_sp_data ()oatlib.sensor.Sensor.ts_zeros

Read Stress period data from table

---------------------------------------------------------------------------------------------------

remove_selected ()oatlib.sensor.Sensor.ts_zeros

Remove selected lake from database

---

create_lak_layer ()oatlib.sensor.Sensor.ts_zeros

create new lake layer and add to qgis

**Parameters:**

- **start_time** (*str*) – starting timestamp of the time serie
- **lenght** (*int*) – lenght of the time serie
- **frequency** (*str*) – frequency of the time serie (‘H’,’D’,’M’,’Y’)

---

get_crs_from_model (*model_name)*oatlib.sensor.Sensor.ts_zeros

get the crs from modeltable_

**Parameters:**

- **model_name** (*str*) – the name oft he selected model

---

clear_sp_table ()oatlib.sensor.Sensor.ts_zeros

Clear stress period lak table and fill with

---

get_sp_period ()oatlib.sensor.Sensor.ts_zeros

get stress period list from timetable

---

layer_list_order(layer_list)oatlib.sensor.Sensor.ts_zeros

Read from the base model the bottom value to automatically set the layer level

**Parameters:**

- **layer_list** (*list*) – te list of qgis layers

---

add_layer_to_group(layer_name, group_id)oatlib.sensor.Sensor.ts_zeros

Load layer from sqlite and add to lake group

**Parameters:**

- **layer_name** (*str*) – the name oft he layer
- **group_id** (*int*) – the id oft he group

---

add_table_to_group (table_name, group_id)oatlib.sensor.Sensor.ts_zeros

Load layer from sqlite and add to lake group

**Parameters:**

- **table_name** (*str*) – the name oft he table
- **group_id** (*int*) – the id oft he group

---

execute_query(query, params=None)oatlib.sensor.Sensor.ts_zeros

execute query on database

**Parameters:**

- **query** (*str*) – the query string
- **params** (*list*) – list of parameter values

---

dd_lake_to_table(params))oatlib.sensor.Sensor.ts_zeros

Add new lake to sqlite table

**Parameters:**

- **params** (*dict*) – dictionary of lake parameters from the interface

---

create_new_lak_layer(new_table_name,base_table_name,layer_id=0)oatlib.sensor.Sensor.ts_zeros

Create a new lak layer based on model layers & the new table & copy the data from base layer

**Parameters:**

- **new_table_name** (*str*) – the name oft he new lake table
- **base_table_name** (*int*) – the table name of base layer
- **layer_id** (*int*) – the layer id

---------------------------------------------------------------------------------------------------

__check_table ()oatlib.sensor.Sensor.ts_zeros

Check if lak table exists

---------------------------------------------------------------------------------------------------

create_sp_lak_table ()oatlib.sensor.Sensor.ts_zeros

create the stress period lake table

---------------------------------------------------------------------------------------------------

message_to_gui(title, message='')oatlib.sensor.Sensor.ts_zeros

Send a messag to the use with an alert box

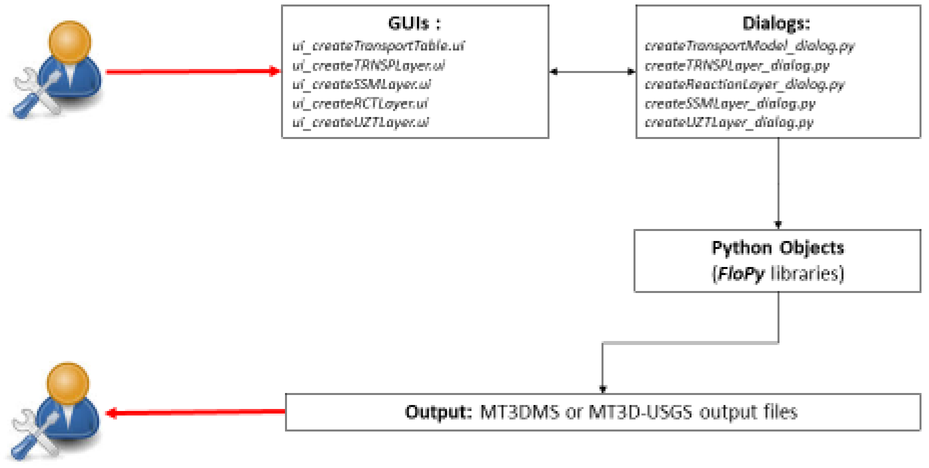
**Parameters:**

- **title** (*str*) – the title oft he alert
- **message** (*int*) – the message tob e shown

# Appendix C Programmer documentation for Solute Transport

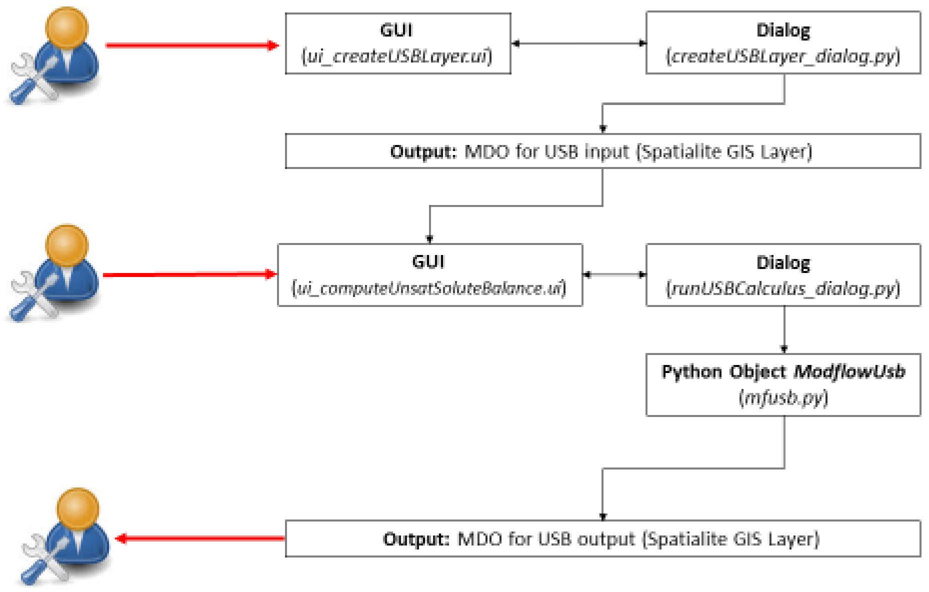
The code organization for the solute transport section reflects the one used for groundwater flow modelling (section above). In particular, the link among different Python files is summarized in the diagram below. Also in this case, an extensive use of Flopy library has been included, to write input files for the code MT3DMS and to read its binary outputs.

The only exception differing from this structure is the part devoted to run USB (Unsaturated Mass Balance), since this module has been specifically developed within FREEWAT plugin, without the inclusion of any external code. For this reason, the USB module is separately documented with more detail in next Appendix.



# Appendix D Programmer Documentation for USB module

USB module is a Python code integrated within FREEWAT plugin. The following schematic summarizes the USB architecture:


GUI
(ui_createUSBLayer.ui)
Dialog
(createUSBLayer_dialog.py)
Output: MDO for USB input (Spatialite GIS Layer)
GUI
(ui_computeUnsatSoluteBalance.ui)
Dialog
(runUSBCalculus_dialog.py)
Python Object ModflowUsb
(mfusb.py)
Output: MDO for USB output (Spatialite GIS Layer)

The core of USB module is the Python object ModflowUSB, whose details are reported in the following section.

## *ModflowUSB* object

### Attributes summary table

| Variable Name | Type/Size | Description |
|---|---|---|
| model | Input/Python object | FloPy model object to which this package will be linked. |
| uzf | Input/Python object | FloPy UZF Package object to which this package will be linked |
| celldim | Input/Scalar | Dimension of the model grid (in units of length selected for the model) |
| cell_dict | Input/Python dictionary | Dictionary of cells where solute source is present.<br>Keys are cell id; values are a list [row,col], where row and col are row and column of that cell, respectively |
| mu_dict | Input/Python | Dictionary of 1st-order degradation factor. |

| | | |
|---|---|---|
| | dictionary | Keys are cell id, values are the degradation factor for cell corresponding to cell id. |
| cin_dict | Input/Python dictionary | Dictionary of concentration at ground surface. Keys are stress periods (0-based); values are an array of shape = (nrow, ncol). |
| cout_dict | Output/Python dictionary | Dictionary of concentration at water table. Keys are stress periods (0-based); values are an array of shape = (nrow, ncol). |

## Methods

The ModflowUsb object consists of 3 methods: write_uzf_gaeges, solute_fluxout, run_model. Explanation is provided below for each method.

### write_uzf_gages

This method takes as input the model structure and the cells of the model grid where the source is present. Therefore, it calls and run the model including in UZF package the ability to write detailed output files for the cells of interest. In this way, for each of these cells, information on unsaturated thickness, infiltration rate at the ground surface and recharge flux at the water table are saved. They will be used in next steps of the process.

*Input*: the ModflowUsb object itself (*self*)

*Output*: no variable in output. At the end of the process new _*.uzf_ output files are written, one for each cell representing the contaminated zone. They are named *modelname.uzfN,* where *N* is the cell id.

*Execution scheme*:

- retrieve UZF package of Modflow model and relative information
- retrieve the cells of interest, and their cell id
- write a gage file for each cell
- update MODFLOW .NAM file and run the MODFLOW model again

### solute_fluxout

This method solve the mass balance difference equation (equation (2) in Section 2.1), for a given cell and for a given stress period.

*Input*:

1. *dt*, stress period length
2. *cin*, concentration at the ground surface
3. *qin*, infiltration flux at the ground surface
4. *z*, unsaturated thickness

5. *th*, porosity
6. *mu*, degradation factor
7. *cold*, concentration at the water table computed at the previous stress period

*Output*:

1. *cout*, concentration at the water table computed for the current stress period

*Execution scheme*:

- Use inputs to compute the concentration, applying formula (2) reported in Section 2.1

### run_model

Run the mass balance calculation for the entire model.

*Input*:

1. *self,* the ModflowUsb object and its attributes
2. *Nsteps*, list of number of time steps, for each stress periods
3. *SPlength*, list of stress period lengths, for each stress periods

*Output*:

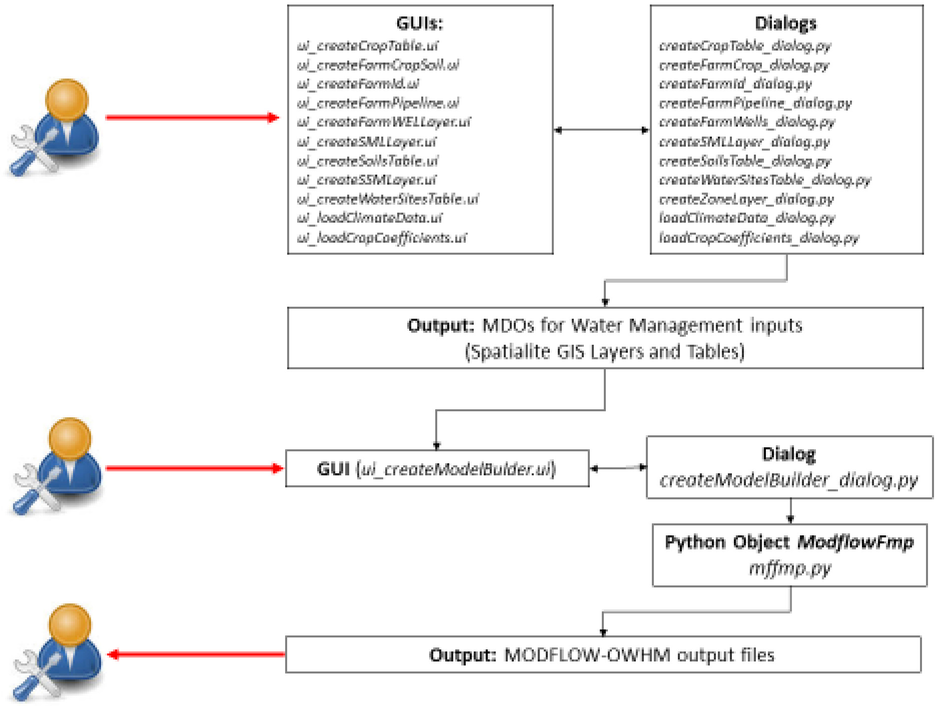
2. *cout_dict*, Python dictionary of concentration at water table.

*Execution scheme*:

- retrieve data from UZF output files
- for each cell and for each stress period, call the method *solute_fluxout* to compute the output
- save the output in the dictionary

# Appendix E Programmer Documentation for Water Management Module

Water Management module consists of a series of Python objects to run the FARM PROCESS capabilities of MODFLOW-OWHM code. The following schematic summarizes the code architecture:


GUIs:
ui_createCropTable.ui
ui_createFarmCropSoil.ui
ui_createFarmId.ui
ui_createFarmPipeline.ui
ui_createFarmWELLayer.ui
ui_createSMLLayer.ui
ui_createSoilsTable.ui
ui_createSSMLayer.ui
ui_createWaterSitesTable.ui
ui_loadClimateData.ui
ui_loadCropCoefficients.ui
Dialogs
createCropTable_dialog.py
createFarmCrop_dialog.py
createFarmId_dialog.py
createFarmPipeline_dialog.py
createFarmWells_dialog.py
createSMLLayer_dialog.py
createSoilsTable_dialog.py
createWaterSitesTable_dialog.py
createZoneLayer_dialog.py
loadClimateData_dialog.py
loadCropCoefficients_dialog.py
Output: MDOs for Water Management inputs
(Spatialite GIS Layers and Tables)
GUI (ui_createModelBuilder.ui)
Dialog
createModelBuilder_dialog.py
Python Object ModflowFmp
mffmp.py
Output: MODFLOW-OWHM output files

The core of this module is the Python object *ModflowFmp*, whose details are reported in the following section. This object is called by the method *buildFmp* defined in the main Python file *createModelBuilder.py.* Method *buildFmp* takes inputs from the QGIS GUI objects and transforms them in the input needed by *ModflowFmp* object.

## *ModflowFmp* object

### Attributes summary table

For details on each attribute, the reader is referred to FarmProcess and MODFLOW-OWHM User's Manuals (Schmid et al., 2006; Schmid et al., 2009;  Hanson et al., 2014).

| Variable Name | Type/Size | Description |
|---|---|---|
| model | Input/Python object | FloPy model object to which this package will be linked. |
| npfwl | Input/integer | Number of farm well parameters |

| | | |
|---|---|---|
| mxl | Input/integer | Maximum number of parameter farm wells |
| mxactw | Input/integer | Maximum number of active farm wells |
| nfarms | Input/integer | Maximum number of water-balance subregions (WDUs) |
| ncrops | Input/integer | Number of crop types |
| nsoils | Input/integer | Number of soil types |
| ifrmfl | Input/integer | Variable Farm ID flag |
| irtfl | Input/integer | Root depth flag |
| icufl | Input/integer | Consumptive-use flag |
| iftefl | Input/integer | Fraction-of-transpiration-and-evaporation-of-crop-consumptive-use flag |
| iieswfl | Input/integer | Fraction-of-inefficiency-losses-to-SW-runoff flag |
| ieffl | Input/integer | Efficiency Flag |
| iebfl | Input/integer | Efficiency Behavior Flag |
| irotfl | Input/integer | Crop rotation flag |
| ideffl | Input/integer | Deficiency Scenario flag |
| iben | Input/integer | Crop-Benefits Flag |
| icost | Input/integer | Water-Cost Coefficients Flag |
| iallotgw | Input/integer | Variable Groundwater Allotment flag |
| iccfl | Input/integer | Concept used for the approximation of ET-fluxes with changing head |
| inrdfl | Input/integer | Non-Routed Surface-Water Delivery Flag |
| mxnrdt | Input/integer | Maximum number of non-routed delivery types |
| isrdfl | Input/integer | Semi-Routed Surface-Water Delivery Flag |
| irdfl | Input/integer | Routed Surface-Water Delivery Flag |
| isrrfl | Input/integer | Semi-Routed Surface-Water Runoff Return Flow Flag |
| Irrfl | Input/integer | Routed Surface-Water Runoff Return Flow Flag |
| iallotsw | Input/integer | Surface-water allotment flag |
| pclose | Input/integer | User specified closure criterion for |

|  |  |  |
|---|---|---|
|  |  | simulated diversions into diversion segments if prior appropriation is chosen |
| ifwlcb | Input/integer | Farm well budget print flags |
| ifnrcb | Input/integer | Farm net recharge budget print flags |
| isdpfl | Input/integer | Farm supply and demand print flags |
| ifbpfl | Input/integer | Farm budget print flags |
| ietpfl | Input/integer | Farm Total Evapotranspiration print flags |
| irtpfl | Input/integer | Optional routing information print flag |
| iopfl | Input/integer | Optional print settings if Acreage-Optimization is chosen |
| ipapfl | Input/integer | Optional print settings if Prior Appropriation is chosen |
| itmp | Input/integer | Flag and counter |
| gse | Input/array | Array (nrow, ncol) of the ground surface elevation |
| farmwells | Input/Python dictionary | Dictionary of farm wells.<br>Keys are stress periods; values are a list [Layer Row Column Farm-Well-ID Farm-ID QMAX] |
| Fid | Input/array | Array (nrow, ncol) of WDU id |
| Ofe | Input/list | List of OFE for each WDU |
| farmcost | Input/Python dictionary | Dictionary of farm costs.<br>Keys are farm id; values are a list of water delivery costs |
| nonrouted | Input/Python dictionary | Dictionary of non-routed deliveries.<br>Keys are farm id; values are a list [FarmID, NRDV , NRDR, NRDU] |
| allotsw | Input/Python dictionary | Dictionary of surface water allotments.<br>Keys are farm id; values are the sw allotment. |
| allotgw | Input/Python dictionary | Dictionary of groundwater allotments.<br>Keys are farm id; values are the gw allotment. |
| srswlocation | Input/Python dictionary | Dictionary of semi-routed deliveries.<br>Keys are farm id; values are the list [Farm-ID Row Column Segment Reach] |

| | | |
|---|---|---|
| Sid | Input/array | Array (nrow, ncol) of soils id |
| soillist | Input/Python dictionary | Dictionary of soil type.<br>Keys are soil id; values are the list [soil name, capillary fringe] |
| cid | Input/array | Array (nrow, ncol) of crops id |
| Root | Input/Python dictionary | Dictionary of root depth.<br>Keys are stress periods; values are the list of root depth for each crop |
| Psi | Input/Python dictionary | Dictionary of potentiometric heads for root uptake process<br>Keys are soil id; values are [PSI1 to PSI4] |
| cropevap | Input/Python dictionary | Dictionary of crop evaporation fraction.<br>Keys are crop id; values are the list [FTR, FEP, FEI] |
| cropfies | Input/Python dictionary | Dictionary of crop inefficiency fractions.<br>Keys are crop id; values are the list [FIESWP, FIESWI] |
| cropirr | Input/Python dictionary | Dictionary of crop non-irrigation flag.<br>Keys are crop id; values are NONIRR flag |
| cropcost | Input/Python dictionary | Dictionary of crop profit parameters<br>Keys are crop id; values are the list [WPF-Slope, WPF-Int , Crop-Price] |
| ifallow | Input/Python dictionary | Dictionary of crop fallow flag.<br>Keys are crop id; values are IFALLOW flag |
| Kc | Input/Python dictionary | Dictionary of crop coefficients<br>Keys are stress periods; values are the list of Kc for each crop |
| etref | Input/Python dictionary | Dictionary of reference evapotranspiration.<br>Keys are stress periods; values are the array (nrow,ncol) of reference ET. |
| precip | Input/list | List of rainfall rate, for each stress period. |

## Methods

The ModflowFmp object consists of 2 methods: *write_in_files, write_file*.

Explanation is provided below for each method.

### write_in_files

This method takes as input the object attributes and write the external text files ( **.in*) that are called by file input writer.

*Input*: the ModflowFmp object itself (*self*)

*Output*: no variable in output. At the end of the process **.in* text files are written under the subfolder *DATA_FMP*, created within the model working folder.

*Execution scheme*:

- retrieve the flags and the input passed to ModflowFmp
- write the text input files, according to the flag specification.

### write_file

This method write the input file "modelname.fmp" needed by MODFLOW-OWHM to run the FarmProcess capabilities. The file is written within the model working folder.

*Input*: the ModflowFmp object itself (*self*)

*Output*: no variable in output. At the end of the process **.fmp* text file is written under the model working folder.

*Execution scheme*:

- retrieve the flags and the input passed to ModflowFmp
- write the **.fmp* according to the flags setting.

# Appendix F Programmer documentation for AkkaGIS

## AkvaGIS code overview

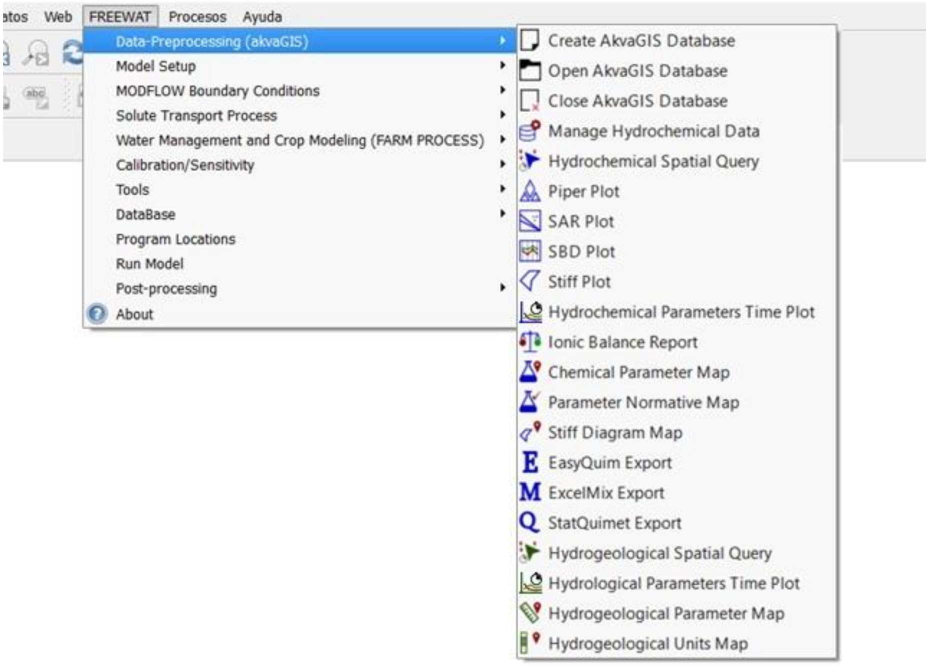
### Overview

The AkvaGIS plugin enhances QGIS with hydrochemical and hydrogeological data processing and analysis. All reference and measurement data is stored in a SQLite database.

The code has been designed following the Object Oriented paradigm. A large effort has been put on avoiding code repetition in order to reduce errors and improve the maintainability

This entry point for the QGIS plugin is the akvaGIS.akvaGIS class. It takes care of adding toolbar buttons to QGIS and linking them to their respective functionality.

The drop-down menu has the following look



The functionality behind each button is called a 'Feature'. Each Feature, with baseclass

feature.AKFeature.AKFeature can either contain other (sub-)Features or/and a form.

There are 3 groups of features:

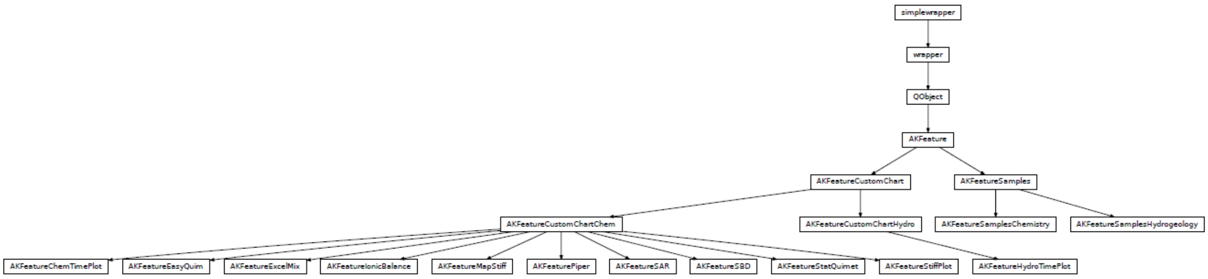
- **General Features.** These are features that deal with the AkvaGIS SQLite database and the related QGIS layers, like create/open/close AkvaGIS database. These features correspond to the group of black colored based toolbar buttons.

- **Hydrochemical Features.** These features manage the selection of geochemical measurements. Different types of plots and maps can be created using the selected samples and measurements. These features correspond to the group of blue colored based toolbar buttons

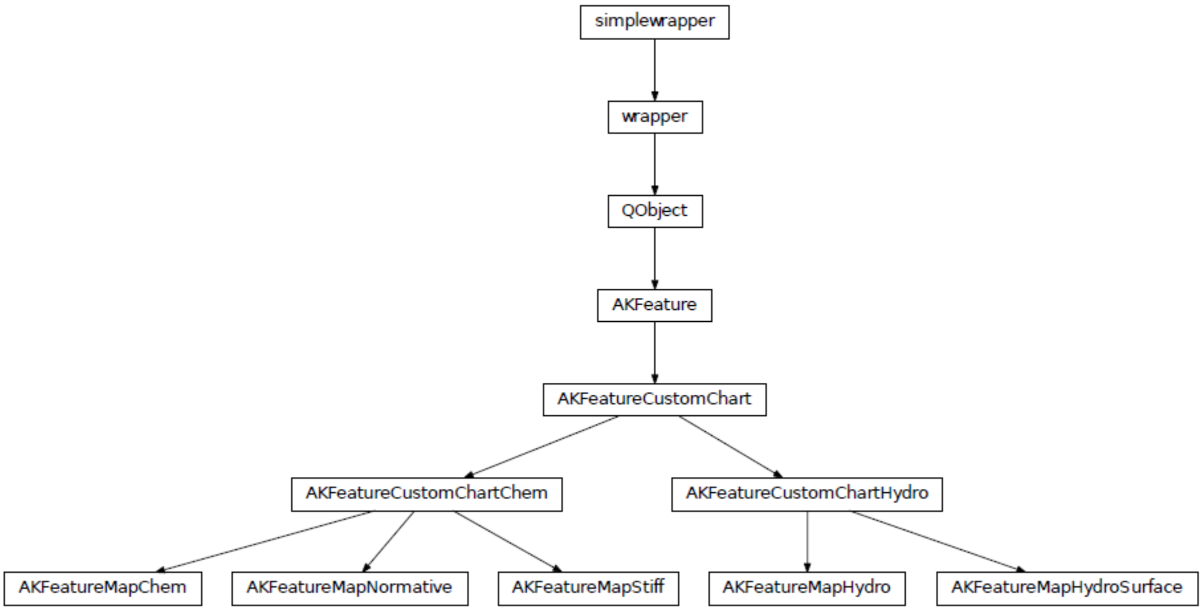
• **Hydrogeological Features.** These features manage the selection of hydrogeological measurements to draw plots and maps. These features correspond to the group of green colored based toolbar buttons.

From the hydrochemical and hydrogeological features there are some to generate plots and others to generate maps:

- These are the features to generate plots:



- And these are the features to generate maps:



All database specific settings are managed by the AKSettings.AKSettings class

All the presented forms are based on the form.AKForm.AKForm class.

All the models used in the (table-)views can be found in the model module.

**Third Party Libraries**

AkvaGIS uses different third party libraries. Some of them are distributed with the plugin (inside the externa 12 module) and others should already be installed in the Python distribution for a correct behaviour of the tool (dependencies).

The following third party libraries are provided with the plugin distribution:

- **ChemPlotLib 1.0**: A GPL licensed library that draws the chemical plots provided in the plugin: Stiff diagram, Piper diagrams and SAR plots but also standard plots as 1D line plots. It relies on Matplotlib 1.5 and. ChemPlotLib offers extensive customization of plots, title labels, axis, edges sizes and colours can be chosen by users.

- **Openpyxl2.3** (https://openpyxl.readthedocs.io): A MIT licensed library for reading and writing Excel 2010 xlsx/xlsm/xltx/xltm files.AkvaGIS uses it to export data to MSExcel spreadsheets format.

- **Odfpy 1.3** (https://pypi.python.org/pypi/odfpy): A library to read and write OpenDocument v. 1.2 files. AkvaGIS uses it to export data to ODF spreadsheets format.

- **Pyexcel 0.2** (https://pyexcel.readthedocs.io): A BSD licensed Python Wrapper that provides one API for reading, manipulating and writing data in csv, ods, xls, xlsx and xlsm files. AkvaGIS uses it to export data to different spreadsheets formats.

The dependencies of the lib are:

- **PyQt4**: The Qt version 4 Python wrapper.

- **Matplotlib 1.5** (http://matplotlib.org/): A Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

## AkvaGis module

**copyright** Copyright (C) 2015 IDAEA-CSIC

**authors** L.M. de Vries, A. Nardi, E. Vázquez Suñé. Velasco Mansilla

**contact** enric.vazquez@idaea.csic.es

**license** GPL General Public License, version 2 or any later version (http://opensource.org/licenses/GPL-2.0)

class akvaGIS (*iface)*

The AkvaGIS plugin enhances QGIS with hydrochemical and hydrogeological data processing and analysis. All reference and measurement data is stored in a SQLite database.

This entry point for the QGIS plugin is the akvaGIS.akvaGIS class. It takes care of adding toolbar buttons to QGIS and linking them to their respective functionality. The functionality behind each button is called a 'Feature'. Each Feature, with baseclass feature.AKFeature.AKFeature can either contain other (sub-)Features or/and a form.

There are 3 groups of features:

- **General Features.** These are features that deal with the AkvaGIS SQLite database and the related QGIS layers, like create/open/close AkvaGIS database.
- **Hydrochemical Features.** These features manage the selection of geochemical measurements. Different types of plots and maps can be created using the selected samples and measurements.
- **Hydrogeological Features.** These features manage the selection of hydrogeological measurements to draw plots and maps.

All database specific settings are managed by the AKSettings.AKSettings class

All the presented forms are based on the form.AKForm.AKForm class.

All the models used in the (table-)views can be found in the model module.

__init__(*iface)*

**Parameters** **iface** (QgsInterface) – An interface instance that will be passed to this

class which provides the hook by which you can manipulate the QGIS application at run time.

**add_feature** (feature=None, parent=None, enabled=False)

Create a QAction and use it for the menu entry and for the toolbar button.

**Parameters**

- **feature** (AKFeature) – The feature provides text, icon and functionality
- **parent** (QObject) – The parent used for the creation of QAction
- **enabled** (bool) – Is enabled on startup or not

initAkvaGIS*(iface)*

initGui ()

Add all the menu entries and toolbar icons to QGIS.

unload ()

Removes the plugin menu item and icon from QGIS GUI.

## AkvaGIS Packages and Modules

**copyright** Copyright (C) 2015 IDAEA-CSIC

**authors** L.M. de Vries, A. Nardi, E. Vázquez Suñé. Velasco Mansilla

**contact** enric.vazquez@idaea.csic.es

**license** GPL General Public License, version 2 or any later version (http://opensource.org/licenses/GPL-2.0)

### AKSettings module

class AKSettings *(parent=None)*

Bases: PyQt4.QtCore.QObject

This class provides the functionality to open and close an AkvaGIS SQLite database. It also takes care of the creation and deletion of related QGIS layers.

**__init__**(parent=None)

If a file called “userSettings.txt” exists in the same directory as this source file, it is used to retrieve the most recently opened database. If not, it is created. The variable self.currentDB points to the database if it is opened, otherwise it is None.

closeCurrentDB ()

getActiveLayerDBPath ()

getCurrentDB ()

getLayerDBPath *(layer)*

Returns path to database directly from the provided layer.

getUserSettings *(settingName)*

Retrieve settings, like the most recently opened database from a text file.

openCurrentDB (*dbFilePath)*

Opens a database and assign it to self.currentDB

openSqliteDatabaseFile (*fileName)*

**setUserSettings** (settingName, settingValue)

Save settings, like the most recently opened database in a text file.

verifyAkvaGISDB ()

Verify if the opened SQLite file is really a AkvaGIS database. Returns True or False.

class TABLES

Any references to actual database tables use the constants defined here. If a table name changes, it only requires an update here.

**CAMPAIGNS** = ‘Campaigns’

**CHEMMEASUREMENTS** = ‘HydrochemicalMeasurements’

**CHEMNORMPARAMS** = ‘HydrochemicalNormativeParameters’

**CHEMPARAMS** = ‘ListHydrochemicalParametersCode’

**CHEMSAMPLES** = ‘HydrochemicalSamples’

**CITATIONS** = ‘Citations’

**HYDROMEASUREMENTS** = ‘HydrogeologicalPointsMeasurements’

**HYDROPARAMS** = ‘ListHydrogeologicalParametersCode’

**HYDROSAMPLES** = ‘HydrogeologicalPointsObservations’

**HYDROUNITS** = ‘HydrogeologicalUnits’

HYDROUNITSAPPEARANCE = ‘HydrogeologicalUnitsApparence’

L_ACTIVITYTYPE = ‘ListActivityType’

L_CAMPAIGNTYPE = ‘ListCampaignType’

L_FUNCTIONCODE = ‘ListFunctionCode’

L_GEOMSRCTYPE = ‘ListGeometrySourceType’

L_HYDROUNITTYPE = ‘ListHydroUnitType’

L_POINTTYPE = ‘ListPointType’

L_RESULTSNATURECODE = ‘ListResultsNatureType’

L_RESULTSQUALITYCODE = ‘ListResultsQualityCode’

L_STATUSCODE = ‘ListStatusCode’

L_UNITOFMEASUREMENTS = ‘ListUnitOfMeasurements’

NORMATIVES = ‘Normatives’

POINTS = ‘Points’

PROCESSES = ‘Processes’

PROJECTS = ‘Projects’

REFSYSTEMS = ‘ReferenceSystems’

RESPONSIBLES = ‘ResponsibleParties’

SAVED_QUERY = ‘AK_SavedQueries’

SAVED_QUERY_HYDRO = ‘AK_SavedQueriesHydro’

SAVED_QUERY_POINTS = ‘AK_SavedQueryPoints’

SAVED_QUERY_POINTS_HYDRO = ‘AK_SavedQueryPointsHydro’

SAVED_QUERY_SAMPLES = ‘AK_SavedQuerySamples’

SAVED_QUERY_SAMPLES_HYDRO = ‘AK_SavedQuerySamplesHydro’

TIMEMETADATA = ‘TimeMetadata’

WELLS = ‘Wells’

WELLS_HYDROUNIT = ‘WellsHydrogeologicalUnit’

### Feature package

### Subpackages

### Feautures.measurements package

### Submodules

### feature.measurements.AKFeatureMeasurements module

**class AKFeatureMeasurements** (settings, iface, windowTitle=u'Query Measurements', parent= None)

Bases: feature.AKFeature.AKFeature

Provides a form to select individual measurements. The selection is for the selected query and a specific set of parameters that can either be selected manually or imposed by the caller of this class. If the measurement value is outside the detection limit range, the limit itself is multiplied by factor that can be set for each base parameter in the listview. Different multiplication factors can be applied if the value is below of above the detection limit. If the value is the result of this multiplication, the column isCalculated will be set to true.

This sub-feature is used by many of the plot and map features.

**__init__**(settings, iface, windowTitle=u'Query Measurements', parent=None)

activateResults()

clearAllAttributes()

**createParameterRow**(tableWidget, values)

deActivateResults()

**findRowOfParameter**(widget, column, value)

getSelectedParameters()

hideColumnsIfNecessary()

imposeParameters(*parameterInfo)*

initialize()

loadAllParametersView()

loadAllParams(query)

loadParameterViews()

**moveParameter**(sourceWidget, destinationWidget, rowNumber) onAddSelectedParameters()

onClose()

onDeselectAllResults()

onLoadQueryView()

onLvQueriesSelectionChanged()

onNext()

onRemoveSelectedParameters()

onRunQuery()

onSelectAllResults()

updateQueryButton()

**nextForm**

## feature.measurements.AKFeatureMeasurementsChem module

**class AKFeatureMeasurementsChem** (settings, iface, windowTitle=u'Query Measurements', parent= None)

Bases: feature.measurements.AKFeatureMeasurements.AKFeatureMeasurements

Make sure that all the queries used by AKFeatureMeasurements reference to the hydrochemical tables.

**__init__**(settings, iface, windowTitle=u'Query Measurements', parent=None)

defineSQLQueries ()

## feature.measurements.AKFeatureMeasurementsHydro module

**class AKFeatureMeasurementsHydro** (settings, iface, windowTitle=u'Query Measurements', parent= None)

Bases: feature.measurements.AKFeatureMeasurements.AKFeatureMeasurements

Make sure that all the queries used by AKFeatureMeasurements reference to the hydrogeological tables.

**__init__**(settings, iface, windowTitle=u'Query Measurements', parent=None)

defineSQLQueries()

hideColumnsIfNecessary()

## Module contents

## feature.results package

### Submodules

### feature.results.AKFeatureResults module

**class AKFeatureResults** (settings, iface, windowTitle=", parent=None)

Bases: feature.AKFeature.AKFeature

This is the base class for the results sub-feature. This feature adds the selected measurements to a new model grouped by sample. Depending on the ResultsFeature-type post-processing is applied to the new model. Some ResultsFeatures require the sample results grouped by parameter, where the values are stored as timeseries. The function storeDataInTimeSeries() takes care of that.

**__init__**(settings, iface, windowTitle=", parent=None)

**addValidColumn**(paramInfo, parameterOrder)

**calcScaledCoordinates**(coordinates, scaleX, scaleY)

**calcTranslatedCoordinatesToPoints**(pointLayer, pointIds, coordinates)

**createUniqueSeriesId**(pointId, parameterId) customizeForm()

getStatistics(*values)*

initialize()

onActionClicked()

onClose()

onSaveTable()

**postProcessModelData**(paramInfo, parameterOrder)

retrieveLocalStiffCoordinates*(valuesByPoint, distY)*

retrieveStiffValues(*valuesByPoint)*

**setModel**(newModel=None)

**showData**(hideMeasurements=False)

storeDataInTimeSeries()

**feature.results.AKFeatureResultsEasyQuim module**

**class AKFeatureResultsEasyQuim**(*settings, iface, parent=None)*

Bases: feature.results.AKFeatureResultsIonicBalance.AKFeatureResultsIonicBalance

This feature provides the functionality for the second EasyQuim Export form.

**__init__**(settings, iface, parent=None)

customizeForm()

onActionClicked()

writeEasyQuimReport*(model)*

**feature.results.AKFeatureResultsExcelMix module**

**class AKFeatureResultsExcelMix**(*settings, iface, parent=None)*

Bases: feature.results.AKFeatureResults.AKFeatureResults

This feature provides the functionality for the second Mix Export form.

**__init__**(settings, iface, parent=None)

customizeForm()

fillMixWidget()

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

**writeMixExcel**(model, timeSeriesData)

**feature.results.AKFeatureResultsIonicBalance module**

**class AKFeatureResultsIonicBalance**(settings, iface, windowTitle=", parent=None)

Bases: feature.results.AKFeatureResults.AKFeatureResults

This feature provides the functionality for the second Ionic Balance form.

**__init__**(settings, iface, windowTitle=", parent=None)

**addValidColumn**(paramInfo, parameterOrder)

customizeForm()

initialize()

**loadData**(model, dataOrder, data)

loadStyles(*textdoc)*

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

writeIonicBalanceReport(*model, textdoc*) feature.results.AKFeatureResultsMap module

**class AKFeatureResultsMap**(settings, iface, windowTitle='Map Results', parent=None)

Bases: feature.results.AKFeatureResults.AKFeatureResults

This feature provides the functionality for the second form to create maps.

**__init__**(settings, iface, windowTitle='Map Results', parent=None)

customizeForm()

getTargetField()

getTargetIndex(*targetField)*

**initialize**(showTable=True)

onActionClicked()

**postProcessModelData**(paramInfo={}, parameterOrder=[])

**feature.results.AKFeatureResultsMapChem module**

**class AKFeatureResultsMapChem**(settings, iface, windowTitle='Map Results', parent=None)

Bases: feature.results.AKFeatureResultsMap.AKFeatureResultsMap

This feature provides the functionality for the second form to create hydrochemical maps.

**__init__**(settings, iface, windowTitle='Map Results', parent=None)

**feature.results.AKFeatureResultsMapHydro module**

**class AKFeatureResultsMapHydro**(settings, iface, windowTitle='Map Results', parent=None)

Bases: feature.results.AKFeatureResultsMap.AKFeatureResultsMap

This feature provides the functionality for the second form to create hydrogeological maps.

**__init__**(settings, iface, windowTitle=’Map Results’, parent=None)

**showData**(hideMeasurements=False)

## feature.results.AKFeatureResultsMapNormative module

**class AKFeatureResultsMapNormative**(settings, iface, windowTitle=’Normative Map Results’,parent=None)

Bases: feature.results.AKFeatureResultsMapChem.AKFeatureResultsMapChem

This feature provides the functionality for the second form to create normative maps.

**__init__**(settings, iface, windowTitle=’Normative Map Results’, parent=None)

**calcParamLimits**(layers, targetFieldIndex)

defineSQLQueries()

**finalizeLayers**(newlayers, targetField, groupNode, paramLimits=None, addLegend=True)

**getNormLimits**(normativeId, paramId)

## feature.results.AKFeatureResultsMapStiff module

**class AKFeatureResultsMapStiff**(settings, iface, windowTitle=’Stiff Chart Map Results’, parent=None)

Bases: feature.results.AKFeatureResultsMapChem.AKFeatureResultsMapChem

This feature provides the functionality for the second form to create stiff maps.

**__init__**(settings, iface, windowTitle='Stiff Chart Map Results', parent=None)

**createLine**(centerX, centerY, distY)

**createPolygon**(coordinates, attributes)

**createStiffPlot**(pointLayer, pointId, coordinates, attributes, distY)

customizeForm()

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

## feature.results.AKFeatureResultsPiper module

**class AKFeatureResultsPiper**(settings, iface, parent=None)

Bases: feature.results.AKFeatureResultsPlot.AKFeatureResultsPlot

This feature provides the functionality for the second form to create piper plots.

**__init__**(settings, iface, parent=None)

customizeForm()

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

## feature.results.AKFeatureResultsPlot module

**class AKFeatureResultsPlot**(settings, iface, plotClass, windowTitle=", parent=None)

Bases: feature.results.AKFeatureResults.AKFeatureResults

This feature provides the base functionality for the second form to create plots.

**__init__**(settings, iface, plotClass, windowTitle=“, parent=None)

customizeForm()

## feature.results.AKFeatureResultsSAR module

**class AKFeatureResultsSAR**(settings, iface, parent=None)

Bases: feature.results.AKFeatureResultsPlot.AKFeatureResultsPlot

This feature provides the functionality for the second form to create SAR plots.

**__init__**(settings, iface, parent=None)

customizeForm()

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

## feature.results.AKFeatureResultsSBD module

**class AKFeatureResultsSBD**(settings, iface, parent=None)

Bases: feature.results.AKFeatureResultsPlot.AKFeatureResultsPlot

This feature provides the functionality for the second form to create SBD plots.

**__init__**(settings, iface, parent=None)

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

### feature.results.AKFeatureResultsStatQuimet module

class AKFeatureResultsStatQuimet(*settings, iface, parent=None)*

Bases: feature.results.AKFeatureResults.AKFeatureResults

This feature provides the functionality for the second form to export data for StatQuimet.

**__init__**(settings, iface, parent=None)

customizeForm()

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

**writeStatQuimetExcel**(model, timeSeriesData)

### feature.results.AKFeatureResultsStiff module

**class AKFeatureResultsStiff**(settings, iface, parent=None)

Bases: feature.results.AKFeatureResultsPlot.AKFeatureResultsPlot

This feature provides the functionality for the second form to create stiff plots.

**__init__**(settings, iface, parent=None)

customizeForm()

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

## feature.results.AKFeatureResultsTimePlot module

**class AKFeatureResultsTimePlot**(settings, iface, windowTitle='TimePlot Results', parent=None)

Bases: feature.results.AKFeatureResultsPlot.AKFeatureResultsPlot

This feature provides the functionality for the second form to create time plots.

**__init__**(settings, iface, windowTitle='TimePlot Results', parent=None)

customizeForm()

initialize()

onActionClicked()

**postProcessModelData**(paramInfo, parameterOrder)

## Module contents

### Submodules

### feature.AKFeature module

**class AKFeature**(iface, parent=None)

Bases: PyQt4.QtCore.QObject

Base Class for AkvaGIS Features

Each AKFeature has the following (optional) functionality:

- initialize()
- run()

This class contains common functionality like:

- saving table data to spreadsheet files
- selecting all/first rows in table views
- generate new file names
- addding QGIS vector layers

**__init__**(iface, parent=None)

**addLabel**(layer, placement=0)

**addLayerToLayersWindow**(layer, parentNode=None)

**addVectorLayer**(layerType, name, projection, fields=[], labelPlacement=1)

closeDatabase()

confirmClosing()

confirmReplacement()

**finalizeLayer**(newLayer, targetField, groupNode, paramLimits=None, labels=[], addLegend=True)

**finalizeLayers**(newlayers, targetField, groupNode, paramLimits=None, labels=[], addLegend=True)

generateNewName*(base)*

getProjection(*layer)*

**groupExists**(*groupName)*

run()

**saveTable**(form, headers, contents)

**selectAllInView**(view, selectMode)

selectFirstInView(*view)*

**feature.AKFeatureCloseDB module**

**class AKFeatureCloseDB**(settings, iface, dialogType, toolbarActions, parent=None)

Bases: feature.AKFeature.AKFeature

**__init__**(settings, iface, dialogType, toolbarActions, parent=None)

enableAkvaGIS*(setEnabled)*

initialize()

updateButtons()

**feature.AKFeatureCreateDB module**

**class AKFeatureCreateDB**(settings, iface, dialogType, toolbarActions, parent=None)

Bases: feature.AKFeature.AKFeature

This feature takes care of creating and loading AkvaGIS databases. When a database is loaded, all the relevant tables are exposed to QGIS by adding layers.

**__init__**(settings, iface, dialogType, toolbarActions, parent=None)

**addDBTableAsLayer**(groupNode, layerName, tableName, geometryColumn)

addDelegatesToLayer()

**addLayer**(parentNode, tableName)

**addNodes**(parentNode, sourceItem)

createLayers()

enableAkvaGIS(*setEnabled)*

initialize()

onBrowse()

onClose()

onCreate()

onOpen()

updateButtons()

**feature.AKFeatureCustomChart module**

**class AKFeatureCustomChart**(settings, iface, windowTitle=“, parent=None)

Bases: feature.AKFeature.AKFeature

This feature takes care of looking up additional information in the database for imposed parameters.

**__init__**(settings, iface, windowTitle=“, parent=None)

addParameterDetails()

defineSQLQueries()

initialize()

**onNextForm**(measurementsModel)

**feature.AKFeatureCustomChartChem module**

**class AKFeatureCustomChartChem**(settings, iface, windowTitle=“, parent=None)

Bases: feature.AKFeatureCustomChart.AKFeatureCustomChart

This base class for all features that are run a query of geochemical measurements.

**__init__**(settings, iface, windowTitle=“, parent=None)

defineSQLQueries()

**feature.AKFeatureCustomChartHydro module**

**class AKFeatureCustomChartHydro**(settings, iface, windowTitle=“, parent=None)

Bases: feature.AKFeatureCustomChart.AKFeatureCustomChart

**__init__**(settings, iface, windowTitle=“, parent=None)

**feature.AKFeatureDBManagement module**

**class AKFeatureDBManagement**(*settings, iface, parent=None*)

Bases: feature.AKFeature.AKFeature

**__init__**(*settings, iface, parent=None)*

**defineSQLQueries**()

**getPointAndUpdateSamples**()

**getPointData**()

**getSampleDetails***(selModel*)

**initialize**()

**loadDataFromFile**(*fileName, tableName, fieldSeparator, skipFirstLine, fieldsContainQuotes,needToInsertValues=False, insertValues=[])*

**loadMeasurementData**(*sampleId)*

**loadMeasurementsFromFile**()

**loadSamplesFromFile**()

**onClose**()

**openEditCampaignForm**()

**openEditMeasurementsForm**()

**openEditPointForm**()

**openEditSampleForm**()

**openEditTableForm**(t*ableName*)

**updateMeasurements**(selected, deselected)

updateSamples(row)

**feature.AKFeatureEasyQuim module**

**class AKFeatureEasyQuim**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Can be used to export the geochemical measurements for the EasyQuim program

**__init__**(settings, iface, parent=None)

**feature.AKFeatureExcelMix module**

**class AKFeatureExcelMix**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Can be used to export the geochemical measurements for the MIX program. The user needs to decide which of the samples are end members before running the export.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureIonicBalance module**

**class AKFeatureIonicBalance**(*settings, iface, parent=None*)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

This feature exports a table with ionic balance information.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureMapChem module**

**class AKFeatureMapChem**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

This feature generates map with hydrochemical measurements information. The user needs to decide if he/she is interested in the first,last,average,minimum or maximum value of all the values for a specific parameter.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureMapHydro module**

**class AKFeatureMapHydro**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartHydro.AKFeatureCustomChartHydro

This feature generates maps for the selected points and parameters

**__init__**(settings, iface, parent=None)

**feature.AKFeatureMapHydroSurface module**

**class AKFeatureMapHydroSurface**(*settings, iface, parent=None*)

Bases: feature.AKFeatureCustomChartHydro.AKFeatureCustomChartHydro

This features generates maps with the top and bottom heights for the selected hydrogeological unit

**__init__**(settings, iface, parent=None)

**createMap**(rowIndex)

defineSQLQueries()

getAllUnitPoints(unitId)

getHydroUnitData()

initialize()

onClose()

onUnitChanged()

onUnitSelected()

**feature.AKFeatureMapNormative module**

**class AKFeatureMapNormative**(*settings, iface, parent=None*)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

**__init__**(settings, iface, parent=None)

defineSQLQueries()

getNormativeData()

getNormativeParameters(normativeId)

initialize()

onNormativeChanged()

onNormativeSelected()

**feature.AKFeatureMapStiff module**

**class AKFeatureMapStiff**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Generate a map with Stiff diagrams. The user first selects the measurements to use and then the general Stiff diagram properties.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureMapPiper module**

**class AKFeaturePiper**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Generate a Piper plot with the selected measurements.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureMapSAR module**

**class AKFeatureSAR**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Generate a SAR plot with the selected measurements.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureMapSBD module**

**class AKFeatureSBD**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Generate an SBD plot with the selected measurements.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureSamples module**

**class AKFeatureSamples**(settings, iface, parent=None)

Bases: feature.AKFeature.AKFeature

This feature takes care of storing a 'query': a selection of points for a specific date-range. Within this selection, specific samples can be deactivated to not form part of the query. Filter functionality is added so it is easier to activate or deactivate a specific set of samples.

**__init__**(settings, iface, parent=None)

activateResults()

clearAllAttributes()

deActivateResults()

deleteStoredQueryDetails()

getIdsOfSelectedPointsInLayer(*cLayer*)

initialize()

loadFilterPointList()

onAddQuery()

onCampaignChanged()

onCampaignEnabled(*checkState)*

onClose()

onDateRangeChanged()

onDateRangeEnabled(c*heckState*)

onDeselectAllPoints()

onDeselectAllResults()

onFilterPointsChanged()

onLoadQueryView()

onLvQueriesSelectionChanged()

onPointsEnabled(*checkState)*

onQuery()

onRefreshSpatialPoints()

onRemoveAllQuery()

onRemoveQuery()

onSelectAllPoints()

onSelectAllResults()

onSpatialSelectionChanged()

updateEditQueryPanel()

updateFilters()

updateTbSamples()

**feature.AKFeatureSamplesChemistry module**

**class AKFeatureSamplesChemistry**(*settings, iface, parent=None)*

Bases: feature.AKFeatureSamples.AKFeatureSamples

Make sure that all the queries used by AKFeatureSamples reference to the hydrochemical tables.

**__init__**(settings, iface, parent=None)

defineSQLQueries()

**feature.AKFeatureSamplesHydrogeology module**

class AKFeatureSamplesHydrogeology(*settings, iface, parent=None)*

Bases: feature.AKFeatureSamples.AKFeatureSamples

Make sure that all the queries used by AKFeatureSamples reference to the hydrogeological tables

**__init__**(settings, iface, parent=None)

defineSQLQueries()

**feature.AKFeatureStatQuimet module**

**class AKFeatureStatQuimet**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Can be used to export the geochemical measurements for the StatQuimet program

**__init__**(settings, iface, parent=None)

**feature.AKFeatureStiffPlot module**

**class AKFeatureStiffPlot**(settings, iface, parent=None)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Generate a Stiff plot with the selected measurements.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureChemTimePlot module**

**class AKFeatureChemTimePlot**(*settings, iface, parent=None*)

Bases: feature.AKFeatureCustomChartChem.AKFeatureCustomChartChem

Generate a time plot with the selected measurements.

**__init__**(settings, iface, parent=None)

**feature.AKFeatureHydroTimePlot module**

**class AKFeatureHydroTimePlot***(settings, iface, parent=None*)

Bases: feature.AKFeatureCustomChartHydro.AKFeatureCustomChartHydro

Generate a time plot with the selected measurements.

**__init__**(settings, iface, parent=None)

**Module contents**

## Form package

### Submodules

### form.AKForm module

**class AKForm**(iface, parent=None)

Bases: PyQt4.QtGui.QDialog

**__init__**(iface, parent=None)

setActiveLayer*(activeLayer)*

### form.AKFormCreatedDB module

**class AKFormCreateDB***(iface, parent=None*)

Bases: form.AKForm.AKForm, Ui_Dialog

**__init__**(iface, parent=None)

### form.AKFormDBManagement module

class AKFormDBManagement(*iface, parent=None*)

Bases: form.AKForm.AKForm, Ui_Dialog

**__init__**(iface, parent=None)

### form.AKFormHydroUnitSelection module

class AKFormHydroUnitSelection(*iface, parent=None)*

Bases: form.AKForm.AKForm, Ui_Dialog

**__init__**(iface, parent=None)

### form.AKFormNormative module

**class AKFormNormative**(*iface, parent=None*)

Bases: form.AKForm.AKForm, Ui_Dialog

**__init__**(iface, parent=None)

### form.AKFormQueryMeasurements module

class AKFormQueryMeasurements(*iface, parent=None)*

Bases: form.AKForm.AKForm, Ui_Form

**__init__**(iface, parent=None)

### form.AKFormQuerySamples module

class AKFormQuerySamples(*iface, parent=None)*

Bases: form.AKForm.AKForm, Ui_Form

**__init__**(iface, parent=None)

**form.AKFormSamplesResults module**

class AKFormSamplesResults(*iface, parent=None*)

Bases: form.AKForm.AKForm, Ui_Form

**__init__**(iface, parent=None)

**Module contents**

**Model Package**

**Submodules**

**model.PlotSttingsModel module**

class PlotSettingsModel(*parent=None*)

Bases: PyQt4.QtCore.QAbstractTableModel

**__init__**(parent=None)

**columnCount**(parent=<PyQt4.QtCore.QModelIndex object at 0x063975B0>)

columnFromField*(family, id)*

**data**(index, role=0)

getDictData()

**rowCount**(parent=<PyQt4.QtCore.QModelIndex object>)

**setData**(index, value, role=2)

setInitialData(*dict_data)*

**model.QueryManagerSamplesModel module**

**class QueryManagerSamplesModel**(*sqlQuery, pkField, parent=None*)

Bases: PyQt4.QtSql.QSqlQueryModel

**__init__**(sqlQuery, pkField, parent=None)

**flags**(index)

reset()

**setActive**(fieldId, value)

setDB(db)

**setData**(index, value, role=2)

**model.SamplesMeasurementsModel module**

class SamplesMeasurementsModel(*parent=None*)

Bases: PyQt4.QtCore.QAbstractTableModel

**__init__**(parent=None)

**addColumn**(parameterId, paramName, baseParam, unit, unitCode, unitId, name, label='')

addFixedColumns()

addSimpleColumn(*header)*

**columnCount**(parent=<PyQt4.QtCore.QModelIndex object at 0x042BE030>)

**data**(index, role=0)

**flags**(*index)*

getCustomColumns()

getMeasurementColumnInfo()

getNumberOfColumns()

**headerData**(section, orientation, role=0)

reset()

retrieveMeasurementColumnParameters(*tableModel)*

retrieveMeasurements(*tableModel*)

**rowCount**(parent=<PyQt4.QtCore.QModelIndex object at 0x042BC3F0>)

**setData**(index, value, role=2)

setInitialData(*tableModel*)

### model.SavedQueryMeasurementsModel module

### class MEASUREMENT_RESULTS

Active = 0

BaseParamId = 8

Campaign = 5

IsCalculated = 16

LimitValue = 11

MeasurementDate = 6

MeasurementId = 11

MeasurementValue = 10

Observation = 3

ParamId = 7

Parameter = 9

Point = 2

PointId = 1

Sample = 3

SampleDate = 4

Unit = 13

UnitCode = 15

UnitId = 14

Value = 12

XCoord = 17

YCoord = 18

count = 19

class SavedQueryMeasurementsModel(*parent=None*)

Bases: PyQt4.QtCore.QAbstractTableModel

**__init__**(parent=None)

**columnCount**(parent=<PyQt4.QtCore.QModelIndex object at 0x042BC470>)

**data**(index, role=0)

**flags**(index)

**headerData**(section, orientation, role=0)

**parseCompValue**(fullString, paramProps)

reset()

**rowCount**(parent=<PyQt4.QtCore.QModelIndex object at 0x042BC3B0>)

setDB(*db)*

**setData**(index, value, role=2)

setOutOfRangeParams(*params*)

**setQuery**(query, pointsTableInfo)

### Module contents

## Resources package

### Submodules

#### resources.resources module

qCleanupResources()

qInitResources()

### Module contents

## Utils package

### Submodules

#### utils.TableUtils module

**addCellInRow**(*row, value)*

addEmptySpacesToRow(*row, numSpaces)*

**addHeader**(textdoc, row, val, numCols, styleName)

valuetype(*val)*

### Module contents

**View package**

**Submodules**

**view.ChecBoxDelegate module**

class CheckBoxDelegate(parent)

Bases: PyQt4.QtGui.QStyledItemDelegate

A delegate that places a fully functioning QCheckBox in every cell of the column to which it's applied

**__init__**(parent)

**createEditor**(parent, option, index)

**editorEvent**(event, model, option, index)

getCheckBoxRect(*option)*

**paint**(painter, option, index)

**setModelData**(editor, model, index)

**Module contents**

**External packages**

## Description

The following third party libraries are provided with the plugin distribution:

- **ChemPlotLib 1.0:** A GPL licensed library that draws the chemical plots provided in the plugin: Stiff diagram, Piper diagrams and SAR plots but also standard plots as 1D line plots. It relies on Matplotlib 1.5 and. ChemPlotLib offers extensive customization of plots, title labels, axis, edges sizes and colours can be chosen by users.

- **Openpyxl2.3** (https://openpyxl.readthedocs.io): A MIT licensed library for reading and writing Excel 2010xlsx/xlsm/xltx/xltm files.AkvaGIS uses it to export data to MSExcel spreadsheets format.

- **Odfpy 1.3** (https://pypi.python.org/pypi/odfpy): A library to read and write OpenDocument v. 1.2 files. AkvaGIS uses it to export data to ODF spreadsheets format.

- **Pyexcel 0.2** (https://pyexcel.readthedocs.io): A BSD licensed Python Wrapper that provides one API for reading, manipulating and writing data in csv, ods, xls, xlsx and xlsm files. AkvaGIS uses it to export data to different spreadsheets formats.

## External module

### extenal package

### Subpackages

### external.chemPlotLib package

### Module contents

### external.odf package

### Module contents

### external.openpyxl package

Module contents

Imports for the openpyxl package.

### external.pyexcel package

### Module contents

### pyexcel

pyexcel is a wrapper library to read, manipulate and write data in different excel formats: csv, ods, xls, xlsx and xlsm. It does not support formulas, styles and charts.

**get_array**(**keywords)

Obtain an array from an excel source

**Parameters keywords** – *see get_sheet()*

get_book_dict*(**keywords)*

Obtain a dictionary of two dimensional arrays

**Parameters keywords –** see get_book()

**get_dict**(name_columns_by_row=0, **keywords)

Obtain a dictionary from an excel source

**Parameters**

- **name_columns_by_row** – specify a row to be a dictionary key. It is default to 0 or first row.
- **keywords** – see get_sheet()

If you would use a column index 0 instead, you should do:

```
get_dict(name_columns_by_row=-1, name_rows_by_column=0)
```

**get_records**(name_columns_by_row=0, **keywords)

Obtain a list of records from an excel source

**Parameters**

- **name_columns_by_row** – specify a row to be a dictionary key. It is default to 0 or first row.
- **keywords** – see get_sheet()

If you would use a column index 0 instead, you should do:

```
get_records(name_columns_by_row=-1, name_rows_by_column=0)
```

# Appendix G Programmer Documentation for OAT

### oatlib.method module

----------------------------------------------------------------------------------------------------

*class* oatlib.method.Methodoatlib.method.Method

Bases: object

Initialize the class

**Parameters: self** (*obj*) – the class

----------------------------------------------------------------

execute(*oat*)oatlib.method.Method.execute

base method for processing

----------------------------------------------------------------------------------------------------

*class* oatlib.method.Compare(*simulation, stats=[u'BIAS'], exponent=1, align=False*)oatlib.method.Compare

Bases: oatlib.method.Method

Calculate comparison statistics of the observation with respect of another serie (simulation)

**Parameters:**

- **simulation** (*oat.Sensor*) – an oat.Sensor object considered as simulation value for comparison.
- **stats** (*list*) – The desired statistics to be avaluated, allowed values are: BIAS, STANDARD_ERROR, RELATIVE_BIAS, RELATIVE_STANDARD_ERROR,
- **COEFFICIENT_OF_EFFICIENCY, INDEX_OF_AGREEMENT, VOLUMETRIC_EFFICIENCY.** (*NASH_SUTCLIFFE,*) –
- **exponent** (*int*) – the exponent used in the calculation of COEFFICIENT_OF_EFFICIENCY or INDEX_OF_AGREEMENT. Allowed values [1, 2]

**Returns:** a dictionary of requested statistics

execute(*oat*)oatlib.method.Compare.execute

execute method

-------------------------------------------------------------------------------------------------

*class* oatlib.method.CumuativeSumoatlib.method.CumuativeSum

Bases: oatlib.method.Method

Returns: a new time series with cumulative sum data

execute(*oat*)oatlib.method.CumuativeSum.execute

-------------------------------------------------------------------------------------------------

*class* oatlib.method.DigitalFilter(*fs*, *lowcut*, *highcut=0.0*, *order=5*, *btype=u'lowpass'*)oatlib.method.DigitalFilter

Bases: oatlib.method.Method

bandpass Butterworth filter

**Parameters:**

- **lowcut** (*float*) – low cutoff frequency
- **highcut** (*float*) – high cutoff frequency
- **fs** (*float*) – sampling frequency
- **order** (*int*) – the filter order
- **btype** (*str*) – band type, one of [‘lowpass’, ‘highpass’, ‘bandpass’, ‘bandstop’]

**Returns:** A new OAT object with filitered data

execute(*oat*)oatlib.method.DigitalFilter.execute

Apply bandpass Butterworth filter to an OAT object

-------------------------------------------------------------------------------------------------

*class* oatlib.method.Exceedance(*values=None*, *perc=None*, *etu=u'days'*, *under=False*)oatlib.method.Exceedance

Bases: oatlib.method.Method

Exceedance probability calculation

**Parameters:**

- **values** (*list*) – list of excedance values to calculate the excedance probability
- **perc** (*list*) – list of exceedance probability to calculate the excedance values
- **etu** (*string*) – excedance time unit, allowed [‘seconds’,’minutes’,’hours’,’days’,’years’], default=’days’

- **under** (*bool*) – caluclate the probability for which values are exceeded (False) or are not exceeded (“True”)

**Returns:** A list of (values,probability,time) tuples, output excedance time is returned according specified *etu* value

execute(*oat*)oatlib.method.Exceedance.execute

----------------------------------------------------------------------------------------------------

*class* oatlib.method.Fill(*fill=None*, *limit=None*)oatlib.method.Fill

Bases: oatlib.method.Method

Different methods for No data filling

**Parameters:**

- **fill** (*str*) – if not null it defines the method for filling no-data optional are: * ‘bfill’ = backward fill * ‘ffill’ = forward fill * ‘time’ = interpolate proportional to time distance * ‘spline’ = use spline interpolation * ‘linear’ = linear interpolation * ‘quadratic’= quadratic interpolation * ‘cubic’= cucbic interpolation
- **limit** (*int*) – if method is ffill or bfill when not null defines the maximum numbers of allowed consecutive no-data valuas to be filled

execute(*oat*)oatlib.method.Fill.execute

Fill no-data

----------------------------------------------------------------------------------------------------

*class* oatlib.method.HydroEvents(*rise_lag*, *fall_lag*, *window=1*, *min_peak=0*, *suffix=u'_event_N'*, *period=None*)oatlib.method.HydroEvents

Bases: oatlib.method.Method

peak flow periods extraction

**Parameters:**

- **rise_lag** (*float*) – The number of days prior to the peak to include in the event hydrograph.
- **fall_lag** (*float*) – The number of days following the peak to include in the event hydrograph.
- **window** (*int*) – Minimum time between successive peaks, in days.
- **min_peak** (*float*) – Minimum value for a peak.
- **suffix** (*string*) – The name of the time series on which statistical calculations will be carried out.

- **period** (*tuple*) – tuple of two elements indicating the BEGIN and END of datetimes records to be used in peak extraction.

**Returns:** A list of oat objects with a storm hydrograph each, they will be named “seriesName+suffix+number” (e.g.: with a series named “TEST” and a suffic “_hyevent_N” we will have: [“TEST_hyevent_N1, TEST_hyevent_N2, ...]

execute(*oat*)oatlib.method.HydroEvents.execute

calculate peak hydrographs

----------------------------------------------------------------------------------------------------

*class* oatlib.method.HydroGraphSep(*mode*, *alpha=0.98*, *bfl_max=0.5*)oatlib.method.HydroGraphSep

Bases: oatlib.method.Method

Perform hydrogram separation

**Parameters:**

- **mode** (*str*) – the method for hydrograph separation.
- **modes are** (*Alleowed*) –
- **TPDF** (***) – Two Parameter Digital Filter (Eckhardt, K., 2005. How to Construct Recursive Digital Filters for Baseflow Separation. Hydrological Processes, 19(2):507-515).
- **SPDF** (***) – Single Parameter Digital Filter (Nathan, R.J. and T.A. McMahon, 1990. Evaluation of Automated Techniques for Baseflow and Recession Analysis. Water Resources Research, 26(7):1465-1473).

**Returns:** a tuple of two oat.Sensor objects (baseflow,runoff)

execute(*oat*)oatlib.method.HydroGraphSep.execute

apply selected mode for hysep

----------------------------------------------------------------------------------------------------

*class* oatlib.method.HydroIndices(*htype*, *code*, *period=None*, *flow_component=False*, *stream_classification=False*, *median=False*, *drain_area=None*)oatlib.method.HydroIndices

Bases: oatlib.method.Method

peak flow periods extraction

**Parameters:**

- **htype** (*str*) – alphanumeric code, one of [MA,ML,MH,FL,FH,DL,DH,TA,TL,TH,RA]

- **code** (*int*) – code that jointly with htype determine the indiced to calculate (see TSPROC HYDROLOGIC_INDECES Table 3-2, page 90)
- **period** (*tuple*) – tuple of two elements indicating the BEGIN and END of datetimes records to be used.
- **flow_component** (*str*) – Specify the hydrologic regime as defined in Olden and Poff (2003). One of [“AVERAGE_MAGNITUDE”, “LOW_FLOW_MAGNITUDE”, “HIGH_FLOW_MAGNITUDE”, “LOW_FLOW_FREQUENCY, HIGH_FLOW_FREQUENCY”, “LOW_FLOW_DURATION”, “HIGH_FLOW_DURATION”, “TIMING”, “RATE_OF_CHANGE”]
- **stream_classification** (*str*) –

  Specify the hydrologic regime as defined in Olden and Poff (2003). One of [“HARSH_INTERMITTENT”, “FLASHY_INTERMITTENT”, “SNOWMELT_PERENNIAL”, “SNOW_RAIN_PERENNIAL”, “

  GROUNDWATER_PERENNIAL”, “FLASHY_PERENNIAL”, “ALL_STREAMS”]
- **median** (*bool*) – Requests that indices that normally report the mean of some other sumamry statistic to instead report the median value.
- **drain_area** (*float*) – the gauge area in m3

**Returns:** A list of oat objects with a storm hydrograph each, they will be named “seriesName+suffix+number” (e.g.: with a series named “TEST” and a suffic “_hyevent_N” we will have: [“TEST_hyevent_N1, TEST_hyevent_N2, ...]

execute(*oat*)

calculate peak hydrographs

---------------------------------------------------------------------------------------------------

*class* oatlib.method.Integrate(*periods=[(None, None)], tunit=u'seconds', factor=1, how=u'trapz', astext=False*)oatlib.method.Integrate

Bases: oatlib.method.Method

Perform integration of time series curve

**Parameters:**

- **periods** (*list*) – a list of tuples with upper and lower time limits for volumes computation.
- **tunit** (*str*) – The time units of data employed by the time series, one of: ‘seconds’, ‘minutes’, ‘hours’, ‘days’, ‘years’.

- **factor** (*float*) – factor by which integrated volumes or masses are multiplied before storage generally used for unit conversion (e.g.: 0.0283168 will convert cubic feets to cubic meters)
- **how** (*str*) – integration method, available methods are: * trapz - trapezoidal * cumtrapz - cumulative trapezoidal * simps - Simpson’s rule * romb - Romberger’s rule
- **astext** – define if dates has to be returned as text (True) or Timestamp (False). Default is False.

**Returns:** a tuple of two oat.Sensor objects (baseflow,runoff)

execute(*oat*)oatlib.method.Integrate.execute

apply selected mode for hysep

----------------------------------------------------------------------------------------------------

*class* oatlib.method.Resample(*freq=u'1Hour'*, *how=None*, *fill=None*, *limit=None*, *how_quality=None*)oatlib.method.Resample

Bases: oatlib.method.Method

Initialize

**Parameters:**

- **freq** (*str*) – Offset Aliases sting (A=year,M=month,W=week,D=day,H=hour,T=minute,S=second; e.g.: 1H10T)
- **how** (*str*) – sampling method (‘mean’,’max’,’min’,first’,’last’,’median’,’sum’), default is ‘mean’
- **fill** (*str*) – if not null it defines the method for filling no-data (‘bfill’= backward fill or ‘ffill’=forward fill), default=None
- **limit** (*int*) – if not null defines the maximum numbers of allowed consecutive no-data valuas to be filled
- **how_quality** (*str*) – sampling method (‘mean’,’max’,’min’,first’,’last’,’median’,’sum’) for observation quality index (default is like ‘how’)

execute(*oat*)oatlib.method.Resample.execute

Resample the data

----------------------------------------------------------------------------------------------------

*class* oatlib.method.SetDataValues(*value, vbounds=[(None, None)], tbounds=[(None, None)]*)oatlib.method.SetDataValues

Bases: oatlib.method.Method

Assign a constant value to the time series

**Parameters:**

- **value** (*float*) – the value to be assigned
- **vbounds** (*list*) – a list of tuples with upper and lower value limits for assignment. bounds are closed bounds (min >= x <= max) e.g: [(None,0.2),(0.5,1.5),(11,None)] will apply: if data is lower then 0.2 # –> (None,0.2) or data is between 0.5 and 1.5 # –> (0.5,1.5) or data is higher then 11 # –> (11,None)
- **tbounds** (*list*) – a list of tuples with upper and lower time limits for assignment. bounds are closed bounds (t0 >= t <= t1)

**Returns:** a new oat.Sensor object with assigned constant value based on conditions

execute(*oat*)oatlib.method.SetDataValues.execute

aaply statistics acording to conditions

----------------------------------------------------------------------------------------------------

*class* oatlib.method.SetQualityStat(*value, vbounds=[(None, None)], tbounds=[(None, None)], stat=u'WT'*)oatlib.method.SetQualityStat

Bases: oatlib.method.Method

Assign a constant weight value to the time series

**Parameters:**

- **value** (*float*) – the value of the weigth to be assigned
- **vbounds** (*list*) – a list of tuples with upper and lower value limits for weigth assignment. bounds are closed bounds (min >= x <= max) e.g: [(None,0.2),(0.5,1.5),(11,None)] will apply: if data is lower then 0.2 # –> (None,0.2) or data is between 0.5 and 1.5 # –> (0.5,1.5) or data is higher then 11 # –> (11,None)
- **tbounds** (*list*) – a list of tuples with upper and lower time limits for weigth assignment. bounds are closed bounds (t0 >= t <= t1)
- **stat** (*str*) – The type of statistics the weight is estimated from, accepted values are: * 'VAR' (Variance) calculated as 1/Statistic * 'SD' (Standard deviation) calculated as 1/(Statistic)^2 * 'CV' (Coefficient of variation) calculated as 1/(Statistic×ObsValue)^2 * 'WT' (Weight) calculated as simple Statistic * 'SQRWT' (Square root of the weight) calculated as Statistic^2

**Returns:** a new oat.Sensor object with assigned constant weightsand weight_stat

execute(*oat*)oatlib.method.SetQualityStat.execute

aaply statistics acording to conditions

----------------------------------------------------------------------------------------------------

*class* oatlib.method.Statistics(*data=True, quality=False, tbounds=[None, None]*)oatlib.method.Statistics

Bases: oatlib.method.Method

Initialize the class

**Parameters:**

- **data** (*bool*) – if True compute statistics of data (default is True)
- **quality** (*bool*) – if True compute statistics of quality (default is False)
- **tbounds** (*list*) – a list or tuple of string (iso856) with upper and lower time limits for statistic calculation. bounds are closed bounds (t0 >= t <= t1)

execute(*oat*)oatlib.method.Statistics.execute

Compute statistics

----------------------------------------------------------------------------------------------------

*class* oatlib.method.Subtract(*sensor*, *align_method=u'mean'*)oatlib.method.Subtract

Bases: oatlib.method.Method

Subtract the values of the provided sensor object

**Parameters:**

- **sensor** (*oat.Sensor*) – an oat.Sensor object to be used to subtract values.
- **align_method** (*str*) – method for alignment of the sensor time serie

**Returns:** an oat.Sensor objectsensor.Sensors

Note

the process join the series and align with respect to the first

execute(*oat*)oatlib.method.Subtract.execute

execute method

----------------------------------------------------------------------------------------------------

*class* oatlib.method.Statistics(*data=True, quality=False, tbounds=[None, None]*)

Bases: oatlib.method.Method

Initialize the class

**Parameters:**

- **data** (*bool*) – if True compute statistics of data (default is True)
- **quality** (*bool*) – if True compute statistics of quality (default is False)
- **tbounds** (*list*) – a list or tuple of string (iso856) with upper and lower time limits for statistic calculation. bounds are closed bounds (t0 >= t <= t1)

execute(*oat*)

Compute statistics

-----------------------------------------------------------------------------------------------------

*class* oatlib.method.Resample(*freq=u'1Hour'*, *how=None*, *fill=None*, *limit=None*, *how_quality=None*)

Bases: oatlib.method.Method

Initialize

**Parameters:**

- **freq** (*str*) – Offset Aliases sting (A=year,M=month,W=week,D=day,H=hour,T=minute,S=second; e.g.: 1H10T)
- **how** (*str*) – sampling method ('mean','max','min',first','last','median','sum'), default is 'mean'
- **fill** (*str*) – if not null it defines the method for filling no-data ('bfill'= backward fill or 'ffill'=forward fill), default=None
- **limit** (*int*) – if not null defines the maximum numbers of allowed consecutive no-data valuas to be filled
- **how_quality** (*str*) – sampling method ('mean','max','min',first','last','median','sum') for observation quality index (default is like 'how')

execute(*oat*)

Resample the data

-----------------------------------------------------------------------------------------------------

*class* oatlib.method.Fill(*fill=None*, *limit=None*)

Bases: oatlib.method.Method

Different methods for No data filling

**Parameters:**

- **fill** (*str*) – if not null it defines the method for filling no-data optional are: * 'bfill' = backward fill * 'ffill' = forward fill * 'time' = interpolate proportional to time distance * 'spline' = use spline interpolation * 'linear' = linear interpolation * ‘quadratic’= quadratic interpolation * ‘cubic’= cucbic interpolation

- **limit** (*int*) – if method is ffill or bfill when not null defines the maximum numbers of allowed consecutive no-data valuas to be filled

execute(*oat*)

## Fill no-data

**oatlib.sensor module**oatlib-sensor-module

*class* oatlib.sensor.Sensor(*name, prop, unit, lat=None, lon=None, alt=None, tz=0, desc=None, freq=None, data_availability=[None, None]*)oatlib.sensor.Sensor

Inits the oat class

| | |
|---|---|
| **Parameters:** | - **name** (*str*) – the name of the sensor (maximum length is 10 characters)<br>- **prop** (*str*) – the observed property<br>- **unit** (*str*) – the unit of measure of the observed property<br>- **lat** (*float*) – the latitude of the station<br>- **lon** (*float*) – the longitude of the station<br>- **alt** (*float*) – the altitude of the station<br>- **tz** (*int*) – the time zone<br>- **desc** (*str*) – the description of the time serie<br>- **ts** (*obj*) – a pandas timeseries object with (time, value) columns et (event time) as time-index and ov (observed values) as value columns<br>- **data_availability** (*list*) – time period of data availability (sensor historical records) |

__repr__(*line=4*)oatlib.sensor.Sensor.__repr__

the repr method

copy()oatlib.sensor.Sensor.copy

Return a deep copy of the OAT object

---------------------------------------------------------------------------------------------------

delete_from_sqlite(*source, name=None*)oatlib.sensor.Sensor.delete_from_sqlite

Delete the oat object from sqlite

| | |
|---|---|
| **Parameters:** | - **source** (*str*) – the sqlite file (including path)<br>- **name** (*list*) – the sensor name to be used |

---------------------------------------------------------------------------------------------------

*classmethod* from_istsos(*service, procedure, observed_property, basic_auth=None, srid=4326*)oatlib.sensor.Sensor.from_istsos

Create the oat class from istSOS

**Parameters:**

- **service** (*str*) – url of the SOS service
- **procedure** (*list*) – sensor name
- **observed_property** (*list*) – observed property name
- **basic_auth** (*tuple*) – touple of username and password - e.g.: (‘utente’,‘123’)

----------------------------------------------------------------------------------------------------

*classmethod* from_sqlite(*source*, *sensor*)oatlib.sensor.Sensor.from_sqlite

Create the oat class from sqliteif not self.data_availability:

begin = self.oat.ts.index.values[0] end = self.oat.ts.index.values[-1]

self.data_availability = [begin, end]

**Parameters:**

- **source** (*str*) – the sqlite file (including path)
- **sensor** (*list*) – sensor name

----------------------------------------------------------------------------------------------------

load_ts(*stype*, ***kwargs*)oatlib.sensor.Sensor.load_ts

Loader method to append new data to an existing sensor

**Parameters:**

- **stype** (*str*) – data source type
- **kwargs** – arguments as per specific module

Note

kwarg depends on the type instantiated, please take a look at specific load methods

----------------------------------------------------------------------------------------------------

period(*astext=False*)oatlib.sensor.Sensor.period

Method to extract the time series upper and lower time limits

**Parameters: astext** (*bool*) – define if outsput should be a tuple of datetime object or text

plot(*data=True*, *quality=False*, *kind=u'line'*, *data_color=u'b'*, *axis=None*, *qaxis=None*)oatlib.sensor.Sensor.plot

plot function

**Parameters:**

- **data** (*bool*) – the sqlite file (including path)
- **quality** (*bool*) – the sensor name to be used
- **kind** (*str*) – kind of plot
- **()** (*qaxis*) – axis for data
- **()** – axis for quality plot

process(*method*)oatlib.sensor.Sensor.process

Method to apply a method for processing by implementing the BEHAVIORAL VISITOR PATTERN

-----------------------------------------------------------------------------------------------------------

save_as_hobfile(*set1*, *set2*)oatlib.sensor.Sensor.save_as_hobfile

save a list of sensors as MODFLOW's HOB input file

-----------------------------------------------------------------------------------------------------------

save_to_sqlite(*source*, *name=None*, *overwrite=False*)oatlib.sensor.Sensor.save_to_sqlite

Save the oat object to sqlite

**Parameters:**

- **source** (*str*) – the sqlite file (including path)
- **name** (*list*) – the sensor name to be used (it shall be unique)

ts_const(*value*, *start_time*, *lenght*, *frequency=None*)oatlib.sensor.Sensor.ts_const

populate time series with constant values

**Parameters:**

- **value** (*float*) – constant value to populate the time serie
- **start_time** (*str*) – starting timestamp of the time serie
- **lenght** (*int*) – lenght of the time serie
- **frequency** (*str*) – frequency of the time serie ('H','D','M','Y')

-----------------------------------------------------------------------------------------------------------

ts_from_csv(*csvfile, sep=u',', timecol=[0], valuecol=1, qualitycol=-1, skiprows=None, comment=u'#', na_values=[], dayfirst=False, strftime=None, freq=None*)oatlib.sensor.Sensor.ts_from_csv

Load data from a CSV file

**Parameters:**

- **csvfile** (*str*) – Either a string path to a file, URL (including http, ftp, and S3 locations), or any object with a read method (such as an open file or StringIO)
- **sep** (*str*) – A delimiter / separator to split fields on. With sep=None, read_csv will try to infer the delimiter automatically in some cases by “sniffing”. The separator may be specified as a regular expression; for instance you may use ‘|s*’ to indicate a pipe plus arbitrary whitespace.
- **timecol** (*list*) – list of column numbers to be used to parse the times of observations e.g. [0,1]
- **valuecol** (*int*) –
- **qualitycol** (*int*) – the column number containing the quality index e.g. 3
- **skiprows** (*int*) – An integer to skip the first n rows (including headers)
- **comment** (*str*) – A character indicating a comment line not to be imported
- **na_values** (*list*) – List of values to be associated with no data value,
- **dayfirst** (*bool*) – Day came before of month?
- **strftime** (*str*) – strftime directive (see http://strftime.org/)

----------------------------------------------------------------------------------------------------

ts_from_dict(*data*)oatlib.sensor.Sensor.ts_from_dict

### Load data from a dict with the following structure:

**Parameters: data** (*dict*) – dict

### Example

data = {

‘time’: [‘2015-12-01T12:00:00’], ‘data’: [12.56], ‘quality’: [100]

}

----------------------------------------------------------------------------------------------------

ts_from_gagefile(*gagefile*, *startdate*, *property=u'stage'*)oatlib.sensor.Sensor.ts_from_gagefile

Load data from a GAGE file output from modflow

**Parameters:**

- **gagefile** (*str*) – a string path to a file of a MODFLOW GAGE input file
- **startdate** (*str*) – isodate starting date (e.g.: ‘2012-11-21T13:20:00+01:00’)
- **property** (*str*) – the name of the observation to be uploaded as defined in the file (default: ‘stage’) accepted values are: Stage, Flow, Depth, Width, Midpt-Flow, Precip., ET, Runoff

----------------------------------------------------------------------------------------------------

ts_from_hobfile(*hobfile*, *startdate*, *hobname*, *disc*, *outhob=None*, *stat=None*)oatlib.sensor.Sensor.ts_from_hobfile

Load data from an hob file output from modflow

**Parameters:**

- **hobfile** (*str*) – a string path to a file of a MODFLOW HOB input file
- **startdate** (*str*) – isodate starting date (e.g.: '2012-11-21T13:20:00+01:00')
- **hobname** (*str*) – the name of the observation to be uploaded as defined in the file (e.g.: 'HOB1')
- **disc** (*list*) – a list of stress period lengths (e.g.: [) or a string path to a file of a MODFLOW discretization input file
- **outhob** (*str*) – a string path to a file a MODFLOW HOB output file (if specified simulated values are uploaded, if not specified observed values are used)
- **stat** (*str*) – a string defining the STAT to be uploaded as quality value of the seri'STATh' or 'STATdd' (applies to MODFLOW-2000 files only)

-----------------------------------------------------------------------------------------------------

ts_from_istsos(*service*, *procedure*, *observed_property*, *offering=None*, *event_time=None*, *spatial_filter=None*, *basic_auth=None*, *freq=None*, *aggregate_function=None*, *aggregate_interval=None*, *qualityIndex=True*)oatlib.sensor.Sensor.ts_from_istsos

Load data from an istsos server

**Parameters:**

- **service** (*str*) – url of the SOS service
- **procedure** (*list*) – sensor name
- **observed_property** (*list*) – observed property name
- **offering** (*str*) – name of the offering - default value is 'temporary'
- **temporalFilter** (*tuple*) – begin and end instant for a between filter - default value None
- **featureOfInterest** (*list*) – name of the feature of interests - default value None
- **spatialFilter** (*list*) – bbox coordinates as a list [minx,miny,maxx,maxy]- default value None
- **basic_auth** (*tuple*) – touple of username and password - e.g.: ('utente','123')
- **aggregate_function** (*str*) – aggregate function, e.g. MAX, MIN, AVG, SUM, default None
- **aggregate_interval** (*str*) – aggregate interval, expressed in iso 8601 duration e.g. "P1DT12H"
- **qualityIndex** (*bool*) – if True istSOS qualityIndex is loaded

-----------------------------------------------------------------------------------------------------

ts_from_listfile(*listfile*, *startdate=None*, *cum=False*, *prop=u'TOTAL'*, *inout=u'IN'*)oatlib.sensor.Sensor.ts_from_listfile

Load data from a listing file output of modflow model:

**Parameters:**

- **listfile** (*str*) – Either a string path to a file, URL (including http, ftp, and S3 locations), or any object with a read method (such as an open file or StringIO)
- **startdate** (*str*) – isodate starting date (e.g.: ‘2012-11-21T13:20:00+01:00’)
- **cum** (*bool*) – use cumulative volumes if True, use time step rates if False
- **prop** (*str*) – the property to be read; one of ‘STORAGE’, ‘CONSTANT HEAD’, ‘WELLS’, ‘RIVER LEAKAGE’, ‘TOTAL’
- **inout** (*str*) – ‘IN’ or ‘OUT’ volumes

-----------------------------------------------------------------------------------------------------------

ts_from_sqlite(*source*, *sql=None*)oatlib.sensor.Sensor.ts_from_sqlite

### Load data from SQLITE

**Parameters:**

- **source** (*str*) – the sqlite file (including path)
- **sql** (*str*) – the sql selecting two fields named time and data - *optional*

-----------------------------------------------------------------------------------------------------------

ts_ones(*start_time*, *lenght*, *frequency=None*)oatlib.sensor.Sensor.ts_ones

### populate time series with one (1) values

**Parameters:**

- **start_time** (*str*) – starting timestamp of the time serie
- **lenght** (*int*) – lenght of the time serie
- **frequency** (*str*) – frequency of the time serie (‘H’,’D’,’M’,’Y’)

-----------------------------------------------------------------------------------------------------------

ts_randn(*start_time*, *lenght*, *frequency=None*)oatlib.sensor.Sensor.ts_randn

### populate time series with random values

**Parameters:**

- **start_time** (*str*) – starting timestamp of the time serie
- **lenght** (*int*) – lenght of the time serie
- **frequency** (*str*) – frequency of the time serie (‘H’,’D’,’M’,’Y’)

-----------------------------------------------------------------------------------------------------------

ts_zeros(*start_time*, *lenght*, *frequency=None*)oatlib.sensor.Sensor.ts_zeros populate time series with zero (0) values

**Parameters:**

- **start_time** (*str*) – starting timestamp of the time serie
- **lenght** (*int*) – lenght of the time serie
- **frequency** (*str*) – frequency of the time serie (‘H’,’D’,’M’,’Y’)

weight(*method*)oatlib.sensor.Sensor.weight

# Appendix H Programmer Documentation for Calibration Module

There are three Python modules used to support model calibration and uncertainty analysis within the FREEWAT plugin. The modules support head observations (mfhob.py), flow gain/loss observations (mfflwob.py), and creation of the main UCODE input file (ucode_in.py).

The core of each of these modules is reported in the following section.

### *ModflowHob* object

Attributes summary table

| Variable Name | Type/Size | Description |
|---|---|---|
| model | Input/Python object | FloPy model object to which this package will be linked. |
| hob | Input/Python object | FloPy HOB Package object to which this package will be linked |

Methods

The ModflowHob object consists of 1 method: write_file. An explanation is provided below for this method.

**write_file**

This method takes as input the hydraulic head observation locations in space (layer, row, column) and time (stress period and time step) and writes a corresponding Modflow HOB file. It also writes corresponding instruction files that can be used by Ucode in the next step.

*Input*: the ModflowHob object itself (*self*)

*Output*: no variable in output. At the end of the process new **.hob* and *.hob_ins output files are written. They are named *modelname.hob and modelname.hob_ins*, for the head output and corresponding instruction file.

*Execution scheme*:

- retrieve head observation information
- write HOB input
- write instruction file input
- update MODFLOW .NAM file

### *ModflowHob* object

Attributes summary table

| Variable Name | Type/Size | Description |
|---|---|---|
| model | Input/Python object | FloPy model object to which this package will be linked. |
| chob | Input/Python object | FloPy CHOB Package object to which this package will be linked |
| gbob | Input/Python object | FloPy GBOB Package object to which this package will be linked |
| drob | Input/Python object | FloPy DROB Package object to which this package will be linked |
| rvob | Input/Python object | FloPy RVOB Package object to which this package will be linked |

Methods

The ModflowHob object consists of 1 method: write_file. An explanation is provided below for this method.

**write_file**

This method takes as input the flow gain/loss observation locations in space (layer, row, column) and time (stress period and time step) and writes a corresponding Modflow CHOB, GBOB, DROB, or RVOB file.  It also writes corresponding instruction files that can be used by Ucode in the next step.

*Input*: the ModflowHob object itself (*self*)

*Output*: no variable in output. At the end of the process new flow observation and corresponding instruction output files are written. They are named *modelname.chob and modelname.chob_ins* for constant head observations and the corresponding instruction file, *modelname.ghob and modelname.gbob_ins*, for general head observations and corresponding instruction file, *modelname.drob and modelname.drob_ins*, for drain observations and corresponding instruction file, and *modelname.rvob and modelname.rvob_ins*, for the river observations and corresponding instruction file.

*Execution scheme*:

- retrieve flow observation information
- write CHOB, GBOB, DROB, or RVOB input
- write corresponding instruction file input
- update MODFLOW .NAM file

**_UcoderWriter_ object**

Methods

The UcodeWriter object consists of 1 method: write_file. An explanation is provided below for this method.

**write_file**

This method takes as input the quantities and user selections for running Ucode.

*Input*: the ModflowHob object itself (*self*)

*Output*: no variable in output. At the end of the process new **._ucode.in* output file is written. It is named *modelname_ucode.in* and controls the execution options of Ucode.

*Execution scheme*:

- retrieve calibration information
- write Ucode input