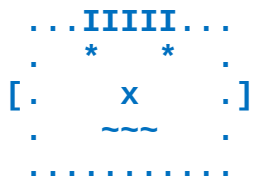
Numerical model for estimating water FLUXes Based On Temperatures-

FLUX-BOT



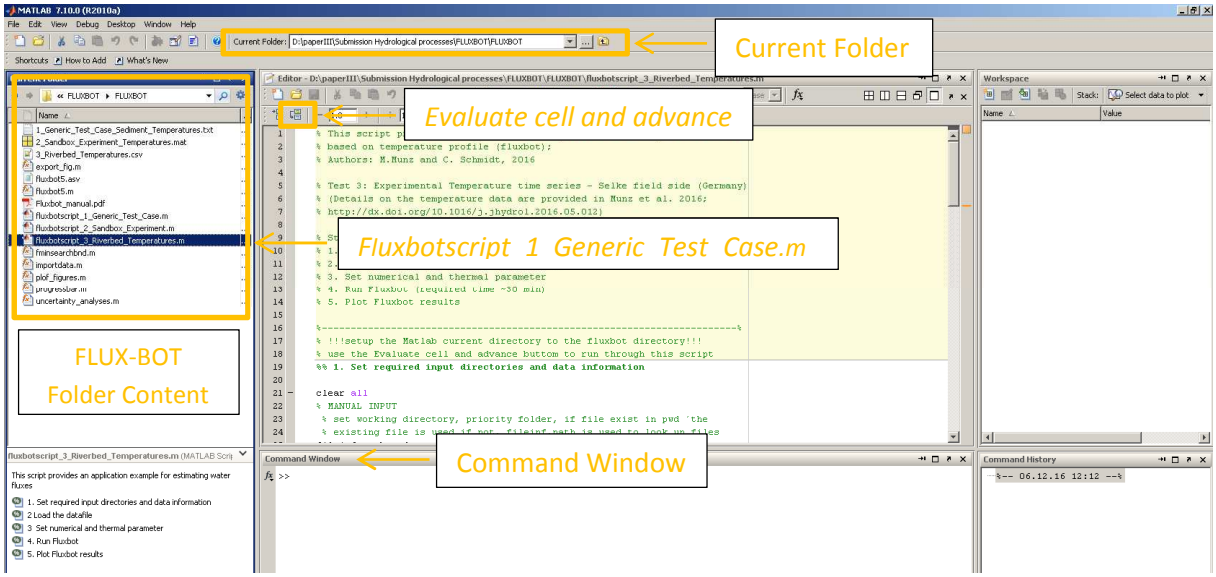
Version 1.0.0

Jan. 05, 2016

Contents

1. FLUX-BOT – example work flow (Quick Start)

1. Download FLUX-BOT (https://bitbucket.org/flux-bot/flux-bot/downloads) to local directory
2. Open MATLAB and set “Current Folder” to your local directory
3. Open Matlab script *fluxbotscript_1_Generic_Test_case.m*
   (Script is displayed in the MATLAB Editor)



4. Run through the matlab script (either by copying the text blocks into the Command Window or by using the *evaluate cell and advance* botton)

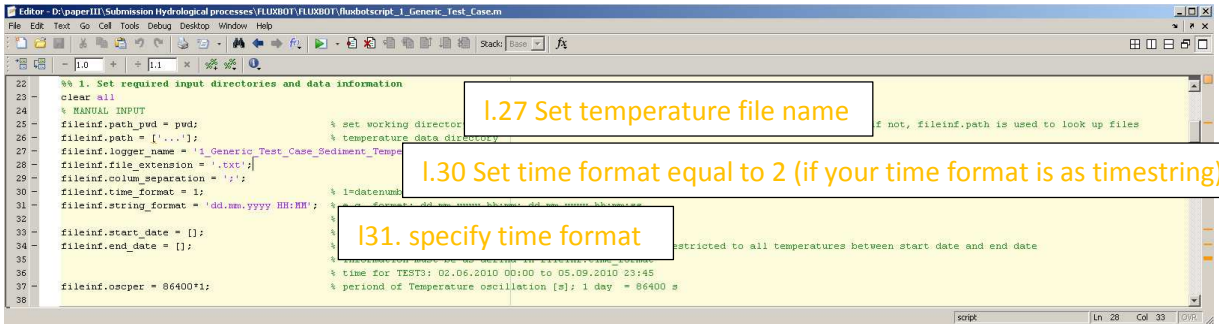
```
1. Set required input directories and data information
2. Load the datafile (Temperature, Depth)
3. Set numerical and thermal parameter
4. Run FLUX-BOT (required time ~2 min)
5. Plot Fluxbot results
```

Adapt example work flow to your application

1. Set required input directories and data information

(if your data are already in MATLAB format continue with 2.)

Copy temperature data into the local directory (FLUX-BOT folder)



Temperature data should be saved as a semicolon-delimited file (.CSV) in the following format:

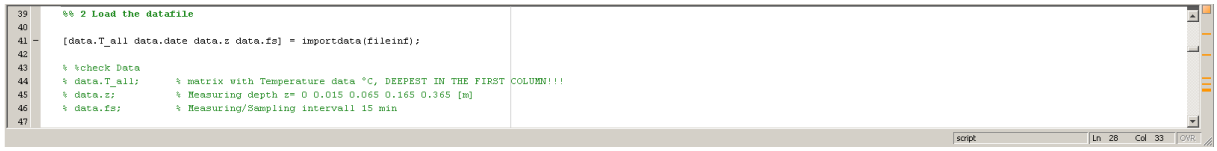
time; 0.61; 0.41; 0.31; 0.26; 0.23; 0.21; 0.06; 0.01; 0

19.06.2012 14:10; 15.37; 15.51; 15.47; 15.64; 15.68; 15.74; 16.12; 16.43

19.06.2012 14:20; 15.37; 15.52; 15.48; 15.68; 15.71; 15.74; 16.19; 16.46

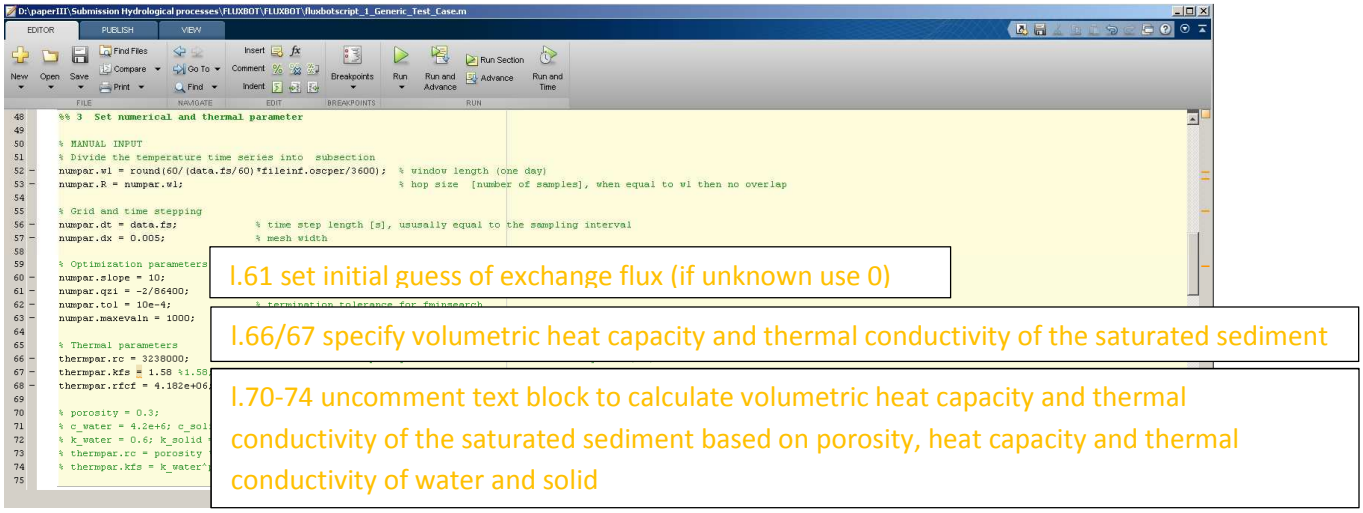
2. Load the datafile (Temperature, Depth)

No manual input required



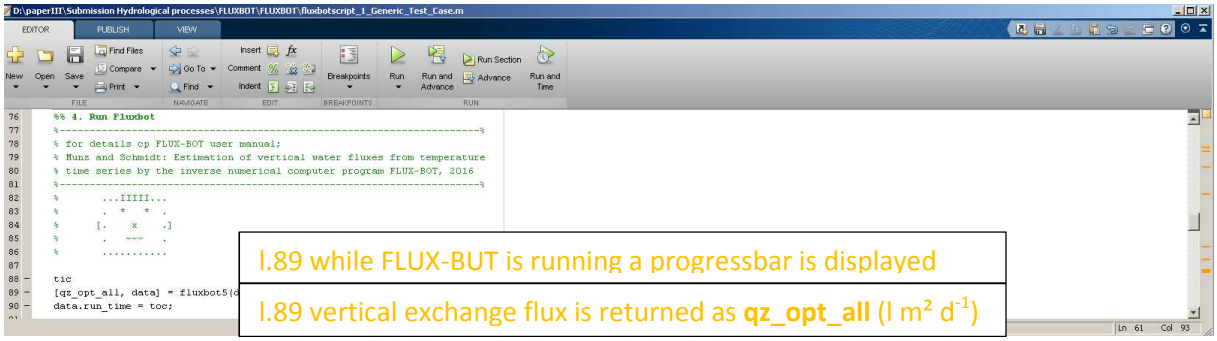
(if your data are in MATLAB format specify them as data.Tall, data.z and data.fs following the data formatting requirements described in the script)

3. Set numerical and thermal parameter



4. Run FLUX-BOT

No manual input required



5. Plot Fluxbot results

No manual input required

IF FLUX-BOT and data pre-processing was successful results are displayed as figures on the screen and saved into the result folder. Exchange flux is written to a text file in the result folder.

Figure 1) Time series of estimated vertical flow velocity

Figure 2) A combined time series plot of the observed and simulated temperatures at the observation depths and the corresponding temperature residuals

Figure 3) A scatter plot of observed versus simulated temperature including the 1:1 line and the quality measures Mean Square Error, Nash-Sutcliffe-Efficiency and Corellation Coefficient for each observation depth not used as boundary condition for numerical vertical flux estimation

2. Model description

FLUX-BOT is a numerical code written in MATLAB that calculates vertical water fluxes in saturated sediments based on the inversion of measured temperature time series. It applies a centered Crank-Nicolson implicit finite difference scheme to solve the one-dimensional heat advection-conduction equation. FLUX-BOT is inspired by Lapham's pioneering work (Lapham, 1989), who applied a numerical solution to the heat flow advection-conduction equation in a forward way.

The motivation to develop FLUX-BOT was the limited applicability of analytical solutions and the software packages they are implemented in, such as VFLUX (Gordon et al. 2012) and EX-STREAM (Swanson and Cardenas, 2011) when the boundary conditions of analytical solutions are not met. A numerical approach offers flexibility in the boundary conditions. This, in turn, reduces the effort of data preprocessing such as the extraction of the diurnal temperature variation from the raw data. A major difference in the flux calculation is that a single vertical water flux is calculated for the entire temperature profile. The analytical methods of Hatch et al. 2006 and Keery et al. 2007 are based on the analysis of pairs of temperature sensors. For FLUX-BOT at least three measurements are required where the shallowest and deepest measuring depths serve as boundary conditions. To ensure robust estimates of vertical water fluxes, their values are estimated and assumed to be constant for a certain period. FLUX-BOT finds the vertical water flux that minimizes the sum of squared differences between measured and modelled temperatures over a user-defined period.

Summary of advantages and limitations:

Advantages:

- Flexibility in the boundary conditions (as all numerical models)
- Minimal effort for preprocessing, e.g. filtering
- The goodness of fit can be derived immediately by comparing measured and modeled temperatures
- Allows sub-daily flux calculation

Disadvantages/Limitations:

- At least three measuring depths are required
- A temperature gradient between surface and sediment temperature must exist
- A single vertical flux is estimated for the entire domain

3. Model implementation

Heat transport at the groundwater- surface-interface (GWSI), conceptualized as water saturated porous medium, can be described by the 1D conduction- advection-equation as proposed by Stallman (1963):

$$\frac{K_{fs}}{\rho c}\frac{\partial^2 T}{\partial z^2} - q_z\frac{\rho_f c_f}{\rho c}\frac{\partial T}{\partial z} = \frac{\partial T}{\partial t}, \quad (1)$$

where K_{fs} is the thermal conductivity of the saturated sediment [Js^{-1} m^{-1} °C^{-1}], T [°C] is the measured temperature at depth z [m] (positive = downward); t is time [s]; q_z is the vertical Darcy flux [L m^{-2} d^{-1}] (positive = downward); ρc is the volumetric heat capacity of the solid - fluid system. ρc can be written as $\rho c = n\rho_f c_f + (1-n)\rho_s c_s$, where $\rho_f c_f$ is the volumetric heat capacity of the fluid $\rho_s c_s$ is the volumetric heat capacity of the solids [J m^{-3} °C^{-1}] and n is the porosity [-]. The magnitude and direction of q_z is a result of Darcy flow, which is controlled by the hydraulic conductivity as the product of the hydraulic conductivity and the total head gradient. In the case of a q_z of zero, Eq. 1 reduces to the Fourier equation of heat conduction.

In FLUX-BOT, Eq.1 is solved numerically by a cell-centered, finite difference Crank-Nicolson numerical scheme. With this, FLUX-BOT calculates the temperatures in each cell as a function of k time increments for a given, constant q_z and time-varying upper and lower temperature boundary conditions. In order to perform the inverse modelling, and estimate q_z from the observed temperatures, the derivative-free Nelder-Mead simplex optimization method, as implemented in Matlab, is applied (Lagarias et al., 1998). The optimization for q_z is not carried out for each observation but for a time window, where q_z is assumed to be constant. If the optimization would be performed for each observation, changes in the observed temperatures, i.e. arising from the diurnal temperature cycle, would be treated as a change in q_z. A typical choice of the window length would be 24 h, where the effect of the diurnal temperature variation would be effectively excluded from the estimation of q_z. To obtain a constant q_z over a certain time window, the objective function to be minimized (E) is represented by:

$$E = \sum_{k=1}^{wl}\sum_{j=1}^{n_z}\left[T_{jk}^{OBS} - T_{jk}^{SIM}\right]^2, \quad (2)$$

where T_{jk}^{OBS} and T_{jk}^{SIM} are the measured and simulated sediment temperatures at depth j over observation depths n_z and for a time window, wl, of length k.

4. Usage

Add the folder containing the FLUX-BOT ("fluxbot") files to the MATLAB search path. The FLUXBOT folder contains all data, scripts and functions needed to run the test examples. If you want to use your own data just add it into the folder (details for the data format and structures are explained in chapter 3).

The script "fluxbotscript_*yourexample*" (e.g. fluxbotscript_1_Generic_Test_Case.m) provides the input variables and can also be used for post-processing (e.g. plotting the data). Check out the verification examples provided and the corresponding matlab test scripts.

To run the Matlab Function fluxbot.m other functions are required. They can be found in the FLUXBOT folder or can be downloaded at the Mathworks page. The function are:

fminsearchbnd.m	Bound constrained optimization (John D'Errico 2006) http://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon
progressbar.m	Figure showing calculation progress (Ohad Gal, 2003) http://www.mathworks.com/matlabcentral/fileexchange/6922-progressbar
export_fig.m	saves a figures or single axes to one or more vector and/or bitmap file formats (Oliver Woodford 2008-2012) http://www.mathworks.com/matlabcentral/fileexchange/23629-export-fig

Data Pre- and Postprocessing functions can be found in the FLUX-BOT folder:

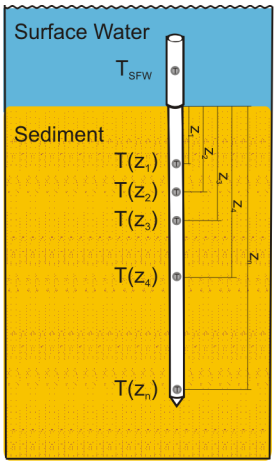
importdata.m	Data import
plot_figures.m	Result plots
uncertainty_analyses.m	Calculation of uncertainty bounds based on a user defined set of Monte Carlo Simulation

For details and the use of these functions see the special sections in the manual.

5. Input/Output data and instructions

Based on a set of temperature time series from a vertical profile and the depth of the corresponding sensors FLUX-BOT calculates the vertical water flow velocity in saturated sediments (all velocities of FLUX-BOT, *qz_opt_all,* are given in l m^{-2} d^{-1}). **Input parameters** for FLUX-BOT are the thermal parameters as volumetric heat capacity (***rc***) and thermal conductivity **(*kfs*)** of the saturated sediment and the volumetric heat capacity of water (normally set to 4.182e+06 J/m³/K). Literature values found for volumetric heat capacity and thermal conductivity of the saturated sediment range between 1 and 8 W m^{-1} K^{-1} and between 1.0e+06 and 4.0e+06 J m^{-3} K^{-1} respectively.
Specific functions are available to evaluate the performance of the temperature simulation (calculation of Mean square error, Nash Sutcliffe Efficiency and Correlation Coefficient and plots of vertical water flow velocity vs. time, Measured and simulated temperature vs. time and scatterplot of observed vs. simulated temperature). FLUX-BOT and all FLUX-BOT functions were developed in MATLAB computing language (The MathWorks, Inc.). The main program is executed in the matlab command prompt, calling required internal functions. All user-defined specifications and parameters are set in the main script **fluxbotscript.m**.

The overall computation time of FLUX-BOT even for long time series with several temperatures per vertical profile is small; e.g. the analyses of temperature time series of several month is done within only a few minutes. No data pre-processing or signal identification is necessary. That makes the code suitable and very easy to apply to even long term temperature time series to estimate vertical water flux in saturated sediments. Beforehand, only the temperature time series should be checked for plausibility and all unrealistic values should be removed. Take care that no NaN is allowed in the temperature time series, FMINSEARCH and **fminsearchbnd**, respectively are not able to handle NaN's.
(FLUX-BOT comes with a function to import typical temperature text files to matlab. The columns of the text file should be as follows: TIMESRING or TIMENUM; TEMP(1); ... ;TEMP(n) in [°C] with one header line: TIME; z(1); ...;z(n) in [m] (Fig. 1). The function is used to import the temperature data, prove for equidistance of all time steps and to remove all columns that include NaN values; in our experience the existence of unrealistic temperature data or NaN values goes along with a failure of the specific temperature sensor, thus the temperature data are not suitable for the analysis)



Time	0	z_1	z_2	z_3	z_4	....	z_n
t_1	$T_{SFW}(t_1)$	$T(z_1,t_1)$	$T(z_2,t_1)$	$T(z_3,t_1)$	$T(z_4,t_1)$	....	$T(z_n,t_1)$
t_2	$T_{SFW}(t_2)$	$T(z_1,t_2)$	$T(z_2,t_2)$	$T(z_3,t_2)$	$T(z_4,t_2)$	....	$T(z_n,t_2)$
....	....	....	....	....	....	....	....
t_m	$T_{SFW}(t_m)$	$T(z_1,t_m)$	$T(z_2,t_m)$	$T(z_3,t_m)$	$T(z_4,t_m)$	....	$T(z_n,t_m)$

e.g.:

Time	0.55	0.35	0.25	0.20	0.15	0
01.01.2011 00:00:00	15.71	16.22	16.05	16.05	15.95	15.88
01.01.2011 00:10:00	…	…	…	…	…	…

Figure 1) Showing possible installation of Multi Level Temperature probe (left) and corresponding Temperature input file for FLUX-BOT (right)

Before presenting the input data needed for FLUX-BOT in detail, some general information about the data and parameter handling is given. All data are declared in data frames to allow an easy transfer between incorporated functions. That allows individual users to add and transfer parameters of interest without any changes in the given program structure. Necessary input data are grouped into four data frames as:

Data: all information of the measured data generated during data import or manual allocations (*data.Tall*, *data.date*, *data.z* and *data.fs*) and all necessary output data

Fileinf: all information to import Temperatures from .txt file (if import function is used)

Thermpar: all thermal parameter

Numpar: all information used to set up the numerical grid and to optimize the vertical flow velocity

5.1. Input data

The required input data for FLUX-BOT can be set by each user individually or generated by the importdata.m function if data are available in standard .txt format (for details of the file structure cp. 1_Generic_Test_Case_Sediment_Temperatures.txt in the FLUX-BOT folder).

Data.T_all: Matrix with the **Temperature time series (°C),** with number of columns equal to the number of temperature time series (DEEPEST TEMPERATURE POSITION MUST BE PLACED IN THE FIRST COLUMN, all others follow with decreasing depth) and with number of rows equal to the number of observation times.

Data.date: vector with all sampling times (formatted as date number) with number of rows equal to the number of sampling times

Data.z: vector with all **sampling depth (m)** with number of columns equal to the number of sampling depths

Data.Fs: **sampling interval (s)**.

5.2. Input parameter

Thermpar.rc: The **volumetric heat** capacity of the saturated sediment (J m^{-3} K^{-1}).

Thermpar.rfcf: The **volumetric heat capacity of water** (J m^{-3} K^{-1}).

Thermpar.kfs: The **thermal conductivity** of the saturated sediment (Jsw^{-1} m^{-1} °C^{-1}).

Numpar.dx: The **vertical mesh width** used to set up the numerical grid. Dx needs to be smaller than the distance between the temperature sensors. We recommend using a mesh width which is ½ to ¼th of the lowest distance between the temperature sensors; for such settings the dependence of the mesh size on the solution was found to be negligible. Note that a mesh refinement significantly increases the calculation time.

Numpar.fs: The **numerical time step**; usually set equal to the temperature sampling interval. If the same time step is used, FLUX-BOT calculates one temperature value for each experimental time step; e.g. all experimental data are used during temperature optimisation process/ for vertical flux estimation.

Numpar.wl: The **window length** (integer) defines the length of a moving window used to analyse the temperature time series/perform vertical flux estimation. Thereby the window

length of one complete day is equal to the number of temperature measurements per day (if *fs* is equal to 600 s, the window length of one day is 144). The vertical flow velocity is calculated as an average over the given window length. The moving window starts at the first time point given in the temperature input time series. Window length should be at least 6-8 hours to include sufficient temperature variation for meaningful temperature optimization/vertical flux calculation (if time step *fs* is equal to 600 s a window length of 48 correspond to a time length of 8 h). For dynamic systems when a fine resolution of vertical flux is needed we recommend to additionally decrease the hop size (*R*) instead of further decreasing the window length. The inspection of scatter plot of observed vs. simulated temperatures indicates appropriate window length. The deviation from the 1:1 line increases with decreasing window length. Note that a reduction in the window length significantly increases the calculation time.

*Numpar.R***:** The **hop size** (integer) defines the actual time shift of the moving window. If hop size is equal to the window length no overlap is set between sequenced windows. The hop size should be set as a portion of the window length. If window length is 48 and hop size is 24, sequent windows overlap for 4 h (if time step is equal to 600 s). To get a high temporal resolution of estimated vertical flow velocities a window length of 8 h and a hop size of 7 h can be used for vertical flux calculation.

The optimisation of temperature residual between observed and simulated temperature is done using the Matlab function **fminsearchbnd** (D'Errico, 2006). Fminsearchbnd is used exactly as the fminsearch function, except that the bounds are applied to the optimization process. Fminsearchbnd finds the minimum of constrained multivariable function using derivative-free method. Besides the optimitzed temperatur this function returns the value of the objective function (*fval_all*) and exit condition of fminsearch (*exitflag_all*). The exit conditions of fminsearch are: 1 = converged, 0 = Maximum number of function evaluations or iterations was reached and -1 = Algorithm was terminated by the output function. Both values are handled back to the main program of FLUX-BOT indicating if the vertical flux estimation was based on a successful temperature optimization. Main function parameter of fminsearchbnd are :

Numpar.qzi: The **initial value** of the vertical flow velocity (**l m^{-2} d^{-1}**). For the first time step the user defined vertical flow velocity is used. For all other time steps the estimated vertical exchange flux of the previous time step is used.

Numpar.Slope: Sets the **lower and upper bound** of the change in qz between subsequent time windows. The specific bounds are calculated based on the initial condition/ vertical exchange flux of the previous window and a user defined slope. *Numpar.Slope* sets the lower and upper bounds required for fminsearchbnd as follows: `qzi±abs(qzi)*slope`. Where `qzi` is either the user defined initial value or the optimized qz from the previous window.

Numpar.tol: The **termination tolerance** for the change in the function value or the change in the estimate of the vertical exchange flux for fminsearchbnd.

Numpar.maxevaln: Maximum **number of function evaluations** allowed for fminsearchbnd.

For *fminsearch* three stopping criteria will be logically combined as (MaxFunEvals OR (TolX AND TolFun)), meaning that reaching *MaxFunEvals* is a hard stopping criterium, while *TolX* and *TolFun* must be both fulfilled to stop the run. If one of the criteria (MaxFunEvals OR (TolX AND TolFun)) is reached fminsearch stops the iteration. Check exitflag for apparant stopping criteria; if 0 increase MaxFunEvals. For further details see the MATLAB Documentation of fminsesarch (The MathWorks, Inc.).

All recommendations for the input parameter settings are based on intense code testing and application to several test cases. Individual setting for specific data sets might be more reliable than the afore mentioned values. The values provided in this manual should give a first idea of parameters to use and should be critically tested and adjusted for each individual application. Variation of most important parameters is easily done in the main program of FLUX-BOT, e.g. no functions need to be changed or updated!

5.3. Output data

If FLUX-BOT execution terminates successfully, the individual results of FLUX-BOT are returned to the matlab workspace/main menu as single vector containing all estimated vertical flow velocities and a combined data frame containing detailed information of the temperature optimisation process. The results of FLUX-BOT are:

qz_opt_all: The **optimized vertical flow velocity** (l m^{-2} d^{-1}) for each window. The corresponding date index is returned as *data.date_ind***.** To get the corresponding time of the input time series use *date(date_ind)*. Vector length of *qz_opt_all* is defined by the length of the original temperature time series devided by the window length.

data.fval_all: The value of the objective function of fminsearchbnd function for each window.

data.exitflag_all: The exit condition of fminsearchbnd for each window. The exit conditions of fminsearch are 1=converged, 0= Maximum number of function evaluations or iterations was reached and -1= Algorithm was terminated by the output function.

data.T_mod_depth: The matrix containing **simulated temperature time series** (delta t is equal to *numpar.fs*, observation time step) for all observation depth. Matrix dimension is length(*data.z*) times length(*data.T_all(:,1)*)

data.date_ind: The **index of central data point of each window**. *Date (data.date_ind)* provides the Date/Time at these.

6. Post calculation - Visualisation function

FLUX-BOT results can be displayed individually using standard matlab plotting functions or by using the visualisation function plof_figures.m. Based on fluxbot.m output data the matlab function plof_figures.m can be run to visualize the results of vertical flow estimation. Three plots are generated:

Figure 4) Time series of estimated vertical flow velocity

Figure 5) A combined time series plot of the observed and simulated temperatures at the observation depths and the corresponding temperature residuals

Figure 6) A scatter plot of observed versus simulated temperature including the 1:1 line and the quality measures Mean Square Error, Nash-Sutcliffe-Efficiency and Corellation Coefficient for each observation depth not used as boundary condition for numerical vertical flux estimation

Data input to the plot function is qz_opt_all and the data data frame (.z, .T_all, .T_mod_depth, .date and .date_ind). Settings of the plof_figures.m are set in the ***plot_info*** data frame as:

.plot_qz ('T'/'F'): if 'T' estimated vertical flow velocity is generated and figures is shown on the screen.

.plot_temperatures ('T'/'F'): same as plot_info.plot_qz for time series of the observed and simulated temperatures at the observation depths and the corresponding temperature residuals

.*scatter_temperatures ('T'/'F')*: same as plot_info.plot_qz for scatter plot of observed versus simulated temperature

.*savefigures ('T'/'F')*: if 'T' created figures will be written as .jpg to the current working directory

.*number_of_scatterpoints (integer between 1 and length(T_all(:,1)))*: sets the number of scatters in the plot. If the plot conitains a large number of scatters it takes a long time to create and to save the plot. Using .number_of_scatterpoints the given number of pairs with highest and pairs with lowest Residuals are plotted.

.*col (vector of the same size than z [number of observation depth])***:** sets the colors of temperature residual time series.

.*fontsize***:** specifying the font size to use for text in units determined by the `FontUnits` property.

7. Post calculation – Uncertainty function

The uncertainty_analysis.m function can be run to calculate the uncertainty bounds of the estimated vertical flow velocity. The uncertainty bounds for the calculated vertical flow velocity resulting from unknown/estimated input parameter (*rc, kfs, T,* and *z*). The uncertainty bounds highlight the impact of changes in the input parameters on the estimation of vertical flow velocity.

To conduct the uncertainty analysis, FLUX-BOT is run a user defined number of times with the parameter sets created by the Monte Carlo sampling, with the uncertain input parameters assigned by multivariate uniform parameter distributions. Uncertainty bounds around the mean flux estimates through time are calculated as two times the standard deviation (2σ) of all FLUX-BOT realisations. The uncertainty bounds represent the uncertainty in a calculated vertical flow velocity due to the often unknown (uncertain) input parameters. All input parameters tested were uncorrelated, so that the selected values for each realisation are unrelated. This provides a standard procedure to compare different flux estimates (in space and time) and check whether they significantly differ from each other or not (Saltelli et al., 2008). Uncertainty analysis was performed for all presented test cases.Besides the input variables used for fluxbot.m this function requires:

MC_info.num_sens_run (integer): number of Monte Carlo model simulation. Take care that overall simulation time is around run time of fluxbot.m times num_sens_run. In our experience around one hundred of Monte Carlo Simulations are sufficient to get a robust estimate of the model sensitivity and uncertainty allowing the calculation of uncertainty bounds even on single core processors in an acceptable time (cp. FLUX-BOT example … and … for details). To

reduce total run time a version making use of the MPI parallelisation is available. Note that the Matlab Parallelisation toolbox is needed.

MC_info.rc_error: uncertainty estimate for the volumetric heat capacity (***rc***). Parameter distribution is defined as normal distribution with mean = rc and standard deviation = rc_error.

MC_info.kfs_error: uncertainty estimate for the thermal conductivity (kfs). Parameter distribution is defined as normal distribution with mean = kfs and standard deviation = kfs_error.

*MC_info.T_error***:** uncertainty estimate for the Temperature (T). Error distribution is defined as normal distribution with mean = 0 and standard deviation = T_error. A Specific error is added to the whole temperature time series for each depth.

*MC_info.*z_error**:** uncertainty estimate for the sensor depth (d). Error distribution is defined as normal distribution with mean = 0 and standard deviation = z_error. A Specific error is added to each sensor depth.

Output of the uncertainty_analysis.m is a ***q_sens*** matrix with dimension of length(*qz_opt_all*) times *MC_info .num_sens_run*. Each column contains the optimized vertical flow velocity based on the MCs input parameter. The corresponding input parameter are transferred to the matlab main menu as *thermpar.rc_sens, thermpar.kfs_sens, thermpar.z_sens* and *thermpar.T_sens*. These results can be used to calculate the mean and std of all estimated vertical flow velocities. Adequate visualisation can be easily added to figure1 created in the plot_figures.m visualisation function.

For the visualisation of uncertainty_analysis.m you can make use of an incorporated plotting function. Therefor *set plot_info.plot_parameter_dist* = 'T' and *plot_info.plot_uncertainty_bounds* = 'T' in the plot_info data frame. Making use of these settings the two following plots are generated while executing the unncertainty_analysis.m function as:

Figure 7) Histograms of the parameter distribution used for the corresponding Monte Carlo simulations of FLUX-BOT. Four subplots are used to visualise the histograms of volumetric heat capacity (absolute value), thermal conductivity (abslute value) , depth (error, added to original depth) and temperarure (error, added to original Temperature);.

Figure 8) Time series of mean estimated vertical flow velocity of all MCs and Time series of mean +- standard deviation of all MCs.

8. Verification example

8.1. Generic Test Case (fluxbotscript_1_Generic_Test_Case.m)

8.1.1. Vertical flow velocity

Table 2) Input parameter for FLUX-BOT

Parameter	Value	Unit
numpar.dx	0.01	m
numpar.dt	600	s
numpar.slope	2	
numpar.qzi	$-2.3148\ 10^{-5}$	$m\ s^{-1}$
numpar.tol	10^{-6}	
numpar.maxevaln	500	
numpar.wl	24	h
rc	3761400	$J\ m^{-3}\ K^{-1}$
kfs	1.58	W m-1 K-1

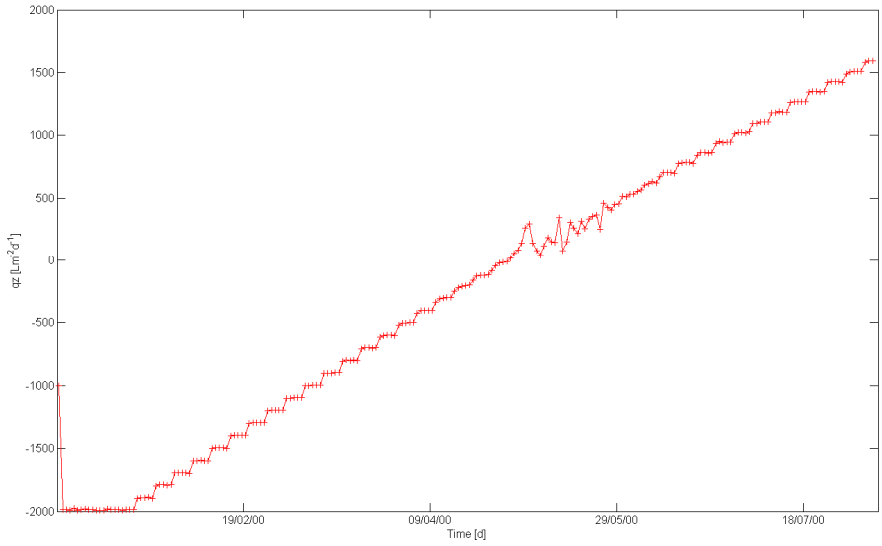


Figure 2) Time series of estimated vertical flow velocity (FLUX-BOT output).

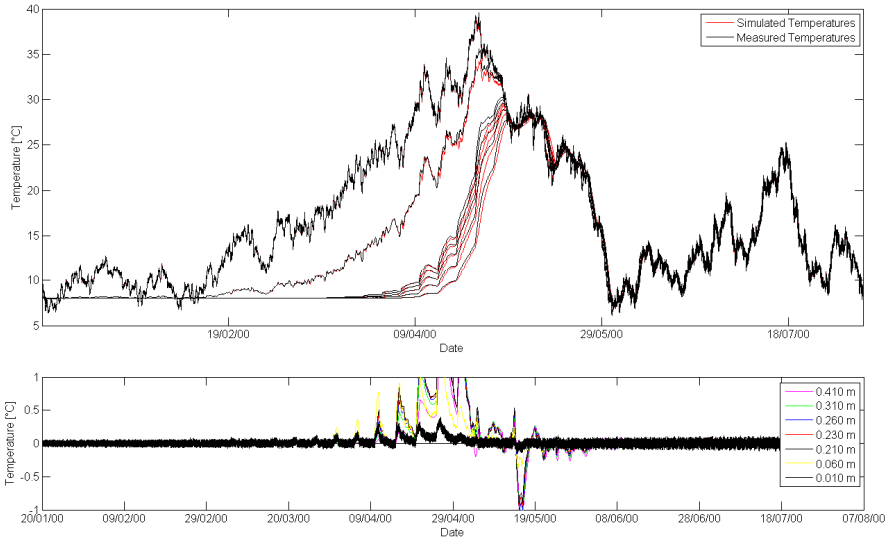


Figure 3) **Time series plot of the observed and simulated temperatures at the observation depths and the corresponding temperature residuals (FLUX-BOT output).**

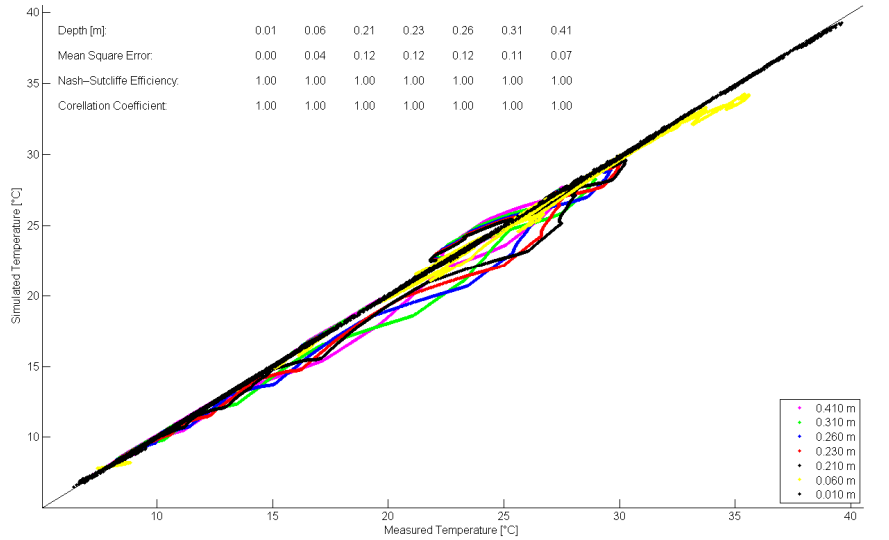


Figure 4) **A scatter plot of observed versus simulated temperature including the 1:1 line and the quality measures Mean Square Error, Nash-Sutcliffe-Efficiency and Corellation Coefficient for each observation depth not used as boundary condition for numerical vertical flux estimation (FLUX-BOT output).**

8.1.2. Uncertainty analyses

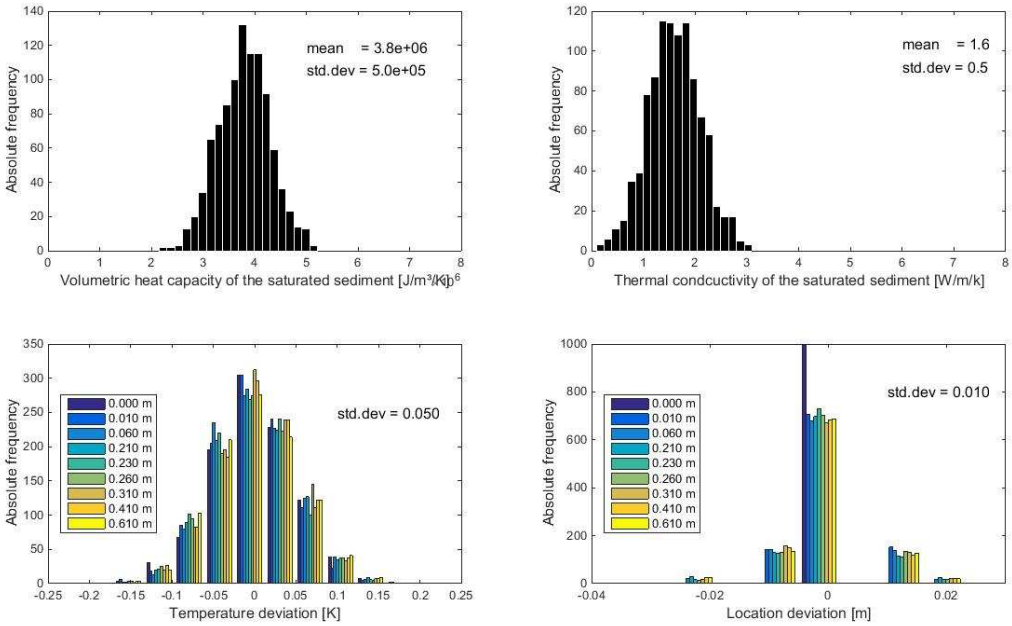


Figure 5) **Parameter space of a) volumetric heat capacity (absolute value), b) thermal conductivity (absolute value) , c) depth (error, added to original depth) and d) temperature (error, added to original Temperature) to perform the uncertainty analyses (FLUX-BOT output).**

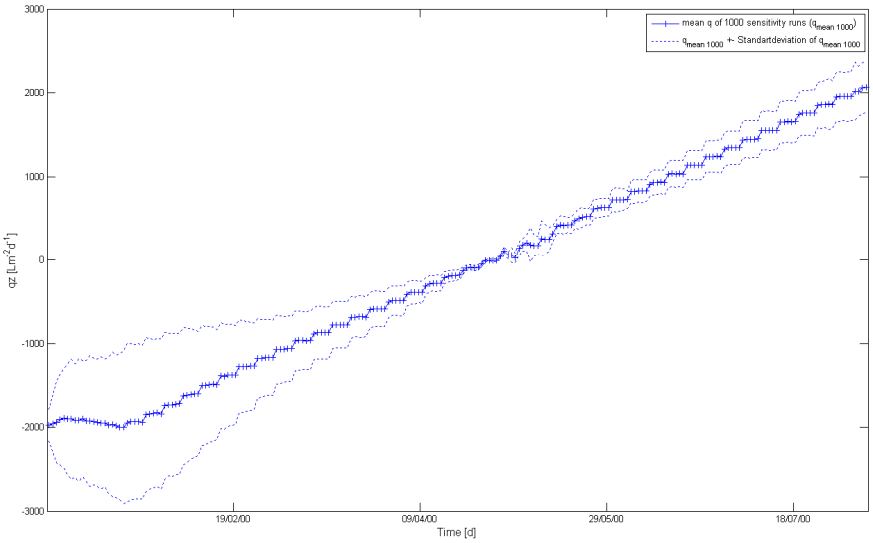


Figure 6) **Time series of estimated vertical flow velocity and upper/lower uncertainty bounds (FLUX-BOT output).**

9. Authors

FLUX-BOT was developed by Matthias Munz[1] and Christian Schmidt[2]

[1] University of Potsdam, Institute of Earth and Environmental Science, Karl-Liebknecht-Str. 24-25 / 14476 Potsdam-Golm / Germany

E-Mail: munz@uni-potsdam.de

[2] Helmholtz Centre for Environmental Research - UFZ, Department Hydrogeology, Permoserstraße 15 / 04318 Leipzig / Germany

Email: christian.schmidt@ufz.de

10. Copyright and License

11. References

Gordon RP, LK Lautz, MA Briggs, JM McKenzie. Automated calculation of vertical pore-water flux from field temperature time series using the VFLUX method and computer program. J Hydrol. 420 (2012) 142-58. http://dx.doi.org/10.1016/j.jhydrol.2011.11.053

Hatch CE, AT Fisher, JS Revenaugh, J Constantz, C Ruehl. Quantifying surface water-groundwater interactions using time series analysis of streambed thermal records: Method development. Water Resour Res. 42 (2006). http://dx.doi.org/10.1029/2005WR004787

Keery J, A Binley, N Crook, JWN Smith. Temporal and spatial variability of groundwater-surface water fluxes: Development and application of an analytical method using temperature time series. J Hydrol. 336 (2007) 1-16. http://dx.doi.org/10.1016/j.jhydrol.2006.12.003

Lagarias JC, JA Reeds, MH Wright, PE Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. Siam J Optimiz. 9 (1998) 112-47. http://dx.doi.org/10.1137/S1052623496303470

Lapham, W. W.: Use of temperature profiles beneath streams to determine rates of vertical ground-water flow and vertical hydraulic conductivity, Report 2337, 1989.

Munz M, SE Oswald, C Schmidt. Sand box experiments to evaluate the influence of subsurface temperature probe design on temperature based water flux calculation. Hydrol Earth Syst Sc. 15 (2011) 3495-510. http://dx.doi.org/10.5194/hess-15-3495-2011

Munz M, SE Oswald, C Schmidt. Analysis of riverbed temperatures to determine the geometry of subsurface water flow around in-stream geomorphological structures. J Hydrol. 539 (2016) 74-87. http://dx.doi.org/10.1016/j.jhydrol.2016.05.012

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S.: Global sensitivity analysis: The primer, WILEY, 2008.

Stallman RW. Computation of groundwater velocity from temperature data. Methods of Collecting and Interpreting Groundwater Data, Washington, DC, 1963. pp. 36 –46.

Swanson TE, MB Cardenas. Ex-Stream: A MATLAB program for calculating fluid flux through sediment-water interfaces based on steady and transient temperature profiles. Comput Geosci-Uk. 37 (2011) 1664-9. http://dx.doi.org/10.1016/j.cageo.2010.12.001