Hydrogeologic Testcase for Noisy Kriging-based Optimization

Short review of problem

1. To be found: Optimal drinking water well locations (x_i and y_i coordinates) and extraction rate(Q_i) for a number of wells, n_{tot}, with i = 1,…n_{tot}.

2. Optimal refers to the maximum total extraction rate $Q_{tot} = \Sigma|Q_i|$ (the absolute values are used since extraction rates are set by negative numbers in the numerical simulation model) under the following constraint:

3. Additional criteria must be fulfilled for a large number of so-called realizations of stochastic hydraulic conductivity fields (For the test case we want at least 99% of all realizations (number of all realizations, n_{real}= 1000)):

    a. A 50-day residence time must be insured for water that is extracted from the river at the left. This is simulated by the numerical flow model in the following way: Particles – representing a small water parcel – that are placed at the river groundwater interface are tracked forward in the direction of flow for 50 days and then stopped. Comparing particle endpositions with well locations the total number of captured river particles, n_{riv}, can be determined. More formally, the first criterion reads: $n_{riv} = 0$, as no particles should be captured by extraction wells.

    b. Deeper groundwater discharges into the Quaternary aquifer from a valley to the right. This groundwater is sulfate rich and only a small portion of it should be mixed into the drinking water that is extracted by the wells in the Quaternary aquifer. Again, this criterion is simulated by the numerical flow model using particle tracking. The number of captured sulfate particles, n_{sulf}, should be lower than 20% of total number of sulfate particles, $n_{sulf,tot}$. The second criterion reads: $100 \times n_{sulf}/n_{sulf,tot} =< 20\ \%$. $n_{sulf,tot} = 495$ for the test case.

    c. Finally, a too large extraction rate of a well at a certain location will result in unrealistic model behavior that is handled by model cells falling dry. In order to stay within realistic limits no model cells should fall dry. The third criterion is given by a binary variable where 1 indicates dry model cells somewhere and 0 no dry model cells.

4. Short intermezzo: The strange coordinate system (y,x) (this stems from the numerical model); Illustration of coordinate system and grid cell indices (in white):

5. Additional constraint: Well placement area. This constraint is essential, because if wells are set close to the model boundaries one could theoretically extract an infinite amount of water without the well falling dry (eventually the sulfate or 50-day criterion would fail, though). It is **important** to note that all well positions represent cell indices (446 rows corresponding to the y-coordinate and 440 columns corresponding to the x-coordinate). They will be rounded

to the next integer by the Matlab function evalModel.m (see below), so that there is a minimum step size of 1 for a new evaluation point of the objective function in terms of well position (the extraction rate will be evaluated as a real number up to floating point precision).

a. Simple constraint version: all wells should stay in or on the rectangle given by x_{min} = 85, x_{max} = 180 (along rows - easting) and y_{min} = 130 y_{max} = 200 (along columns - southing).

b. More complex version: The left boundary should follow the river – hence all wells should stay within the trapezoid given by the four points (in counterclockwise order)

   i. P1 (y=130,x=85)

   ii. P2 (y=200,x=55)

   iii. P3 (y=200,x=180)

   iv. P4(y=130,x=180)

The well placement area constraint can either be dealt by constraining the search space accordingly or a penalty term.

6. Where does the noise come in? The flow field is tremendously influenced by the spatial arrangement of hydraulic conductivity values. Hydraulic conductivity can vary from about 10^{-1} m/s (coarse gravel) up to 10^{-9} m/s (clay) depending on the sedimentary deposits present. Since aquifers are inherently heterogeneous, i.e. the deposited materials do vary from place to place, local information on hydraulic conductivity (e.g. from boreholes) is typically only valid within a certain neighborhood of the borehole. This is why hydraulic conductivity fields are usually modeled as spatially-correlated random fields. The easiest approach is to model the log-transformed (natural logarithm of base e) hydraulic conductivities as a random Gaussian field with known mean, variance, and spatial correlation lengths. For the simulation of hydraulic conductivity we suggest the following parameters:

   a. Constant known mean μ = -9.2103 (log-transformed - corresponds to 1 x 10^{-4} m/s)

   b. Exponential variogram (γ) model: $\gamma = \sigma^2[1-\exp(-|h|/a)]$, where σ = 2.0 is the (constant) standard deviation of the Gaussian field (log-transformed values!), h is the separation distance or lag and a is the range or correlation length.

   c. The correlation length in x direction (easting) should be set to a_x = 25 m and in y-direction (northing) to a_y = 35 m

   d. The rectangular grid resolution should be 5 x 5 m with 446 rows (southing) and 440 columns (easting).

   e. If needed: hydraulic conductivity realizations can be provided.

Technical details

The archive *HydroGeoTestCase.zip* contains numerous files and a couple of batch scripts and Windows executables. All of them are needed in order to set a certain extraction scenario (locations of extraction wells and according extraction rates), run the numerical simulation and gather

information in terms of the three criteria given above. Interaction with optimization algorithms is, however, intended to be performed via Matlab function calls (which will perform system calls to run the simulation etc.). Since all necessary files are included in the archive it is advisable to set the Matlab command line directory to the folder where you run your optimization problem:

1. Matlab functions (contain also a short documentation within the source code):

    a. evaluateRealn.m: Probably the most important function. Its signature reads as follows: [fitn]=evaluateRealn(x,n) where x is a ($3*n_{tot}$) x 1 column vector containing x_i, y_i and Q_i (Q_i need to be negative numbers; i = 1,…n_{tot}) in the order (x_1,y_1,Q_1,x_2,y_2,Q_2,…, xn_{tot},yn_{tot},Qn_{tot})^T; n $\in$ {1,…,n_{real}} is the number of the realization that should be evaluated, and fitn is a 1 x 3 row vector containing (n_{riv}, n_{sulf}, dry) for the particular realization n. IMPORTANT: The function expects that all realizations are named *field.** where * represents one particular n $\in$ {1,…,n_{real}} and that they contain hydraulic conductivities in m/s (!). It will then copy *field.** to the file *myField.dat* which will be read into the numerical model once the simulation is invoked. In case you want to have full control of the field generation yourself, you need to make sure that the one hydraulic conductivity field you want to evaluate is stored in the file *myField.dat* before you call the numerical model routines (details which are covered within the next matlab function).

    b. evalModel.m: Is the function that actually places the wells, sets extraction rates, calls the numerical model routines and reads the output by use of batch scripts and system calls to Windows executables. Its signature is [n_{riv},n_{sulf},dry]=evalModel(x), where x is a ($3*n_{tot}$) x 1 column vector containing x_i, y_i and Q_i (Q_i need to be negative numbers; i = 1,…n_{tot}) in the order (x_1,y_1,Q_1,x_2,y_2,Q_2,…, xn_{tot},yn_{tot},Qn_{tot})^T. It returns (n_{riv}, n_{sulf}, dry) for whatever realization/hydraulic conductivity field is stored in *myField.dat*.

    c. objFunc.m: Shows an example implementation of the objective function to be minimized. In the particular example objFunc.m is used to set n_{real}, $n_{sulf,tot}$ and the cut-off percentage for the allowable number of sulfate particles. The constraints are handled by penalty terms which are added to the total sum of extraction rates, $Q_{tot} = \Sigma|Q_i|$. The exact formulation of the penalty terms in objFunc.m can be called ‘naïve’ –however- it is functional in combination with the CMA-ES algorithm (the running of the CMA-ES example requires runCMAES.m to be called from the Matlab command prompt).

    d. evaluateAllReals.m: Alternative function which will evaluate all n_{real} hydraulic conductivity realizations in one go and return the n_{tot} x 3 matrix fitn containing (n_{riv}, n_{sulf}, dry) for each realization (this function is called by objFunc.m for the example case with n_{real} = 5).

2. New realizations: In order to build hydraulic conductivity realizations on-the-fly depending on your algorithm needs we summarize the important information:

    a. Random Gaussian field realized on a 5 x 5 m rectangular grid represented by a 446 x 440 (rows x columns) matrix of hydraulic conductivity values given in m/s.

b. The following field parameters apply for the log-transformed hydraulic conductivities: mean μ = -9.2103 (log-transformed - corresponds to 1 x 10^{-4} m/s). Exponential variogram model, standard deviation of Gaussian field σ = 2.0 (of the log-transformed values), correlation length in x direction (easting) should be set to a_x = 25 m and to a_y = 35 m in y-direction (northing).

c. If the Matlab function evaluateRealn is used all fields need to be in stored in ASCII files following the naming convention *field.1, field.2, field.3,…field.99, field.100, …, field.999, field.1000, field.1001* etc. where the number after the point '.' is the index of the realization. If you do not want to use evaluateRealn (or evaluateAllReals) you can copy the field you want to evaluate to a file named *myField.dat* which is the file that is read by the numerical simulation model.

d. Note: The numerical simulation model is, unfortunately, quite sensitive to the format of the input file *myField.dat*. Ideally your field generator works with Fortran90 ASCII formats and uses a 'G' format and prints out the field matrix from north to south (top to bottom) row-wise with 440 columns in one row. All numbers should be separated by whitespace and the floating point format should only contain two digits in the exponent (excluding the sign of the exponent), e.g.

5.12345678E-07 3.12345678E-05 5.12345678E-07 3.12345678E-05
3.12345678E-05 5.12345678E-07 3.12345678E-05 7.12345678E-02
for a 2 x 4 matrix.

e. Apart from the specifications under d) we also provide 10 example fields (files *field.1, field.2, …, field.10*) which work for the numerical simulation model and which can be investigated with respect to the number of newline and carriage return characters (again if more than 10 example fields are required – we can offer a large number of realizations upon request).

3. Objective function statement: For at least 99% of all n_{real} = 1000 realizations strictly keep all three criteria, i.e. n_{riv} = 0, $n_{sulf}/n_{sulf,tot}$ =< 0.2, dry = 0, and constrain well positions to a) the rectangle given by x_{min} = 85, x_{max} = 180 (along rows!) and y_{min} = 130 y_{max} = 200 (along columns - southing) or b) to the trapezoid given by the four points (in counterclockwise order) P1 (y=130,x=85), P2 (y=200,x=55), P3 (y=200,x=180), P4(y=130,x=180) and maximize total extraction rate $Q_{tot} = \Sigma|Q_i|$ (Q_i need to be negative numbers) by adaptation of both, well positions and extraction rates for i) one, ii) two or iii) three wells (probably not more than five wells).

4. Final statement: Ignore all output that might appear on the Matlab command line, unless you see an error statement in English indicating numerical simulation model failure.