Calibration of groundwater models
Draft June 2013

Geohydrology 2

June 19, 2013

1 Introduction

No matter how carefully models are constructed, they generally need to be calibrated. That is, a set of user-chosen model parameters has to be optimized such that the deviation between measurements comply as well as possible with the model outcomes. There are many reasons why outcomes of a model initially deviate significantly from the measurements. For example, lack of accurate data. Differences between the scale of the model and the reach or scale of the measurements, both in time and space. Measurement errors should not be one of them, as the expectation of such errors tend to be or should be zero. Having said that, models also deviate from the measurements, because their concept differs from reality. This may be due to lack of insight in the geology as well as the processes involved. If the latter is the case, calibration may lead to the right outcomes for the wrong reasons. That is, the calibration process blindly optimizes model parameters in order to reduces the difference between model and measurements to a minimum, based on the wrong conceptual model, causing parameters to assume non-plausible values as trying to reduce the errors at the cost of other than the right parameters. Also, if the recharge has been wrongly estimated, the calibration will try to compensate that by adapting the conductivities in the model, which it is often perfectly abel to do so, as the head has tendency to depend on N/k instead of N or k separately. Without an independent measurement of the recharge, N, it may be impossible to obtain unique values for the conductivity. So, generally, calibration assumes that the model concept is correct, and that the chosen parameters are representative of the model. Different model concepts may have to be included or should be included in the calibration process, but this is seldom done, due to cost. Another problem with model calibration is too little data for too many parameters. This will cause the problem to be undefined or badly defined, so that the estimated parameter values will be highly uncertain, despite the model looking quite correct. All these pitfalls, that may yield a calibrated model that replicates the measurements quite well, will shoot you in the foot when the model is used for calibration, as the probability of producing a reliable prediction will be less, the greater the uncertainties in the parameters that have been included in the calibration. Prediction, in fact the aim of the model, has may also suffers from the fact that often it depends on other parameters than those included in the calibration. Next to all this, predictions may be required for circumstances that the model has not met in the past and, therefore, have not played a role in the calibration. This renders predictions unreliable. Often, parameters that were not very important in the calibration, that is, their uncertainty tends to remain high, may very well op paramount importance for the outcomes of prediction to be made by the model by the customer. Hence calibration and use of the model for predictions later on need careful evaluation.

2 Theory

Let us assume we have a set of measurements y and a a model that computes the values and the quantities represented by the measurements, whatever the measurements are (heads, flows, concentrations, ...). We may assume that the computed values in the measured locations depend linearly on the user-chosen parameter vector p at least when $p = p_{opt}$ and if we do not deviate too much from our initial estimate:

$$y_i = y_{p_0 i} + J_{i1}\left(p_{01} + \Delta p_1\right) + J_{i2}\left(p_{02} + \Delta p_2\right) + ...J_{in}\left(p_{0n} + \Delta p_n\right) + \epsilon_i$$

$$\Delta y_i = J_{i,1}\Delta p_1 + J_{i,2}\Delta p_2 + ...\epsilon$$

Where p_0 is our initial parameter set and y_0 the computed output for this parameter set. $\Delta p_i = p_i - p_{0i}$. Any such linear system is writable as

$$\Delta y = J\Delta p + \epsilon \tag{1}$$

Where y are the $n_m \times 1$ vector with the measurements, such as heads, etc, and p the n_p vector with the model parameters that the user choose to include in the calibration. J is an $n_m \times n_p$ coefficient array. J is the sensitivity matrix, also called the Jacobian.

$$J = \frac{\partial \hat{y}}{\partial p} \tag{2}$$

Each element in J, i.e. J_{ij} is the sensitivity of the quantity computed at measurement location i for a change of parameter j. Notice that the Jacobian is independent of the actual measurements, it is only dependent on measurement locations and times.

Each of the n_p columns of J is a sensitivity vector of the computed quantities $\hat{y}$ with respect to one parameter. That is, column i of J has n_m values as follows:

$$J_i = \frac{\partial \hat{y}}{\partial p_i} \tag{3}$$

It is clear that we assume here (eq: 1) that the sought quantity depends linearly on the parameters. This is generally not the case in real models, not even in linear models. But it is always approximately true if we allow only a small variations of the parameters. But as long as the parameters values are incorrect, and the relation between model-computed quantities and the model parameters is not linear, the derivative or sensitivities J depend on the parameter values, causing equation 1 to be incorrect. Therefore, calibration will be done stepwise, by continuously improving J such that the remaining errors ϵ between the model results and the measurements will be minimized.

Notice that we can always compute the sensitivity matrix or Jacobian with our model. Doing this is fundamental. It is generally done by computing y for the parameters at hand once and then n_p times, in which each parameters is changed by some some value in turn. Then

$$J_i = \frac{\hat{y}_{p_i+\Delta p_i, p_{j\neq i}} - \hat{y}_p}{\Delta p_i}$$

which requires $n_p + 1$ model runs for each of the J_i columns of the Jacobian. Sometimes, a somewhat more accurate form is used

$$J_i = \frac{\hat{y}_{p_i+\Delta p_i, p_{j\neq i}} - \hat{y}_{p_i-\Delta p_i, o_{j\neq i}}}{\Delta p_i}$$

which requires $2n_p - 1$ model runs to compute the full Jacobian.
It is also possible to use the adjoint state method, but this is generally too complex for use in general practices, because it requires to be built into the model code, whereas the previous methods can be done with the existing model code without any change. These methods are used by the well known programs that can optimize parameters of any model, i.e. UCODE and PEST.

Hence the objective of the calibration is to minimize the difference between model results and measurements. Taking the classical least squares approach as the simplest example, we want to optimize the values of the model parameters such that the minimize the cost function I, which is here the sum of the squares of the deviation of the measurements and the model:

$$miimize\ \ I = \sum_{i=1}^{N} \epsilon_i^2$$

where I is the cost function, also called objective function. In vector form:

$$I = (\Delta y - J\Delta p)^T (\Delta y - J\Delta p)$$

For convenience we now drop the Δ in front of y and p, but we have to remain aware of the meaning of these parameters.

Writing this out with Δ dropped yields

$$\begin{aligned} I &= y^T y - y^T Jp - (Jp)^T y + (Jp)^T Jp \\ I &= y^T y - y^T Jp - p^T J^T y + p^T J^T Jp \end{aligned}$$

2.1 Side step: differentiation of matrix equations

The matrix A and be expressed as a_{ji} where j is the column index and i the row index. A^T is the equivalent to a_{ij} with row and column interchanged.

The derivative of $A^T x$ with respect to x can be seen as follows. It is equivalent to $\sum_{j=1} a_{ij} x_j$. We'll do this by example

$$A = \begin{bmatrix} a & b & c \\ r & s & t \\ u & v & w \end{bmatrix}, \; A^T = \begin{bmatrix} a & r & u \\ b & s & v \\ c & t & w \end{bmatrix}, \; p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}, \; y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\begin{aligned} Ap &= \begin{bmatrix} ap_1 + bp_2 + cp_3 \\ rp_1 + sp_2 + tp_3 \\ up_1 + vp_2 + wp_3 \end{bmatrix} \\ y^T Ap &= y_1 \{ap_1 + bp_2 + cp_3\} + y_2 \{rp_1 + sp_2 + tp_3\} + y_3 \{up_1 + vp_2 + wp_3\} \end{aligned}$$

$$\begin{aligned} \frac{\partial y^T Ap}{\partial p} &= \begin{bmatrix} \{y_1 a + y_2 r + y_3 u\} & \{y_1 b + y_2 s + y_3 v\} & \{y_1 c + y_2 t + y_3 w\} \end{bmatrix} \\ &= A^T y \end{aligned}$$

and

$$\begin{aligned} A^T y &= \begin{bmatrix} ay_1 + ry_2 + uy_3 \\ by_1 + sy_2 + vy_3 \\ cy_1 + ty_2 + wy_3 \end{bmatrix} \\ p^T A^T y &= p_1 \{ay_1 + ry_2 + uy_3\} + p_2 \{by_1 + sy_2 + vy_3\} + p_3 \{cy_1 + ty_2 + wy_3\} \\ \frac{\partial p^T Ay}{\partial p} &= \begin{bmatrix} \{ay_1 + ry_2 + uy_3\} & \{by_1 + sy_2 + vy_3\} & \{cy_1 + ty_2 + wy_3\} \end{bmatrix} \\ &= A^T y \end{aligned}$$

But if we just focus on the multiplication of vector $\mathbf{a}_i^T x$ we get $\mathbf{a}_i^T = \sum_{j=1} a_{ij} x_j$ differentiating this with respect to x_k just yields a_{ik}. If we do this for all k with $1 \leq k \leq n_p$ we get the vector $\mathbf{a}$ (not transposed). Hence $\partial a_i^T \partial x = a_i$ and because this works the same for all vectors that constitute the matrix A we have

$$\frac{\partial A^T x}{\partial x} = A^T$$

The derivative of $x^T A$ with respect to x and be seen in a similar way. The coefficients in each column a_j are $\sum x_i a_{ij}$. Taking the derivative with respect to x_k of this sum yields a_{kj} which is exactly the column j. Hence for all columns

$$\frac{\partial x^T A}{\partial x} = A^T$$

The derivative of the quadratic form $x^T Ax$ is also important. Written out this form is

$$x^T Ax = \sum\sum x_j a_{ji} x_i$$

Notice that matrix A is symmetric for this multiplication to be possible. Hence summation $i = 1..n$, $j = 1..n$. If we want to differentiate with respect to x_k we need to assemble all coefficients that contain x_k. These are

$$x_k^2 a_{kk}, \;\; x_k a_{kj} x_j, \;\; x_i a_{ik} x_k$$

where $k \neq j$ in the second form and $k \neq i$ in the third. These components in all their combinations constitute exactly 2 times the array A

so

$$\frac{\partial x^T Ax}{\partial x} = 2Ax$$

With this we can compute the derivative of the equation above to obtain a system with n linear equations in n unknowns.

2.2 Finishing the derivation

Differentiation with respect to p, using the results from the box in which it was shown that $\frac{\partial p^T J^T y}{\partial p} = \frac{\partial y^T Jp}{\partial p} = J^T y$ then yields

$$\begin{aligned} \frac{\partial I}{\partial p} &= 0 - 2J^T \Delta y + 2J^T J \Delta p \\ J^T J \;\; \Delta p &= J^T \Delta y \end{aligned}$$

so that

$$\begin{aligned} \Delta p &= \left(J^T J\right)^{-1} J^T \Delta y \\ p &= p_0 + \left(J^T J\right)^{-1} J^T \left(y - y_0\right) \end{aligned}$$

$J^T J$ is a square matrix and J^T has the number of rows equal to the number or parameters and the number of columns equal to the number of measurements.

In Matlab, you can solve a set of equations as follows

$$\begin{aligned} Ap &= y \\ y &= A \backslash p \end{aligned}$$

In Matlab you solve an over-determined system, like he have done above here, namely with more equations than unknowns in the same was

$$\begin{aligned} y &= \left(A^T A\right)^{-1} A^P p \\ y &= A \backslash p \end{aligned}$$

This will always yield the least squares solution.

We can, therefore, immediately compute the parameters if we have the Jacobian J. But the answer can only be true if the relation between computed quantities and parameters were linear, which is not generally the case. Therefore, the differences between our model results and the measurements will be larger than the mere measuring errors. The aim is to sequentially improve the parameter vector so that in the end the remaining errors are minimal.

3 Example test model

Let us assume the model is a groundwater system between two ditches with a given precipitation on it.

Continuity:

$$\frac{dq}{dx} = N \rightarrow q = q_0 + Nx$$

Considering the cross section to the left and the right of the boundary between the two conductivity regions with different length of k_L and k_R:

$$\begin{aligned} q &= q_0 + Nx &= -\frac{k_L}{2}\frac{dh^2}{dx}, \quad x \leq \alpha L \\ q &= q_0 + Nx &= -\frac{k_R}{2}\frac{dh^2}{dx} \quad x \geq \alpha L \end{aligned}$$

$$\begin{aligned} q_0 x + \frac{1}{2}Nx^2 &= C_L - \frac{1}{2}k_L h^2, \quad x \leq \alpha L \\ q_0 x + \frac{1}{2}Nx^2 &= C_R - \frac{1}{2}k_R h^2, \quad x \geq \alpha L \end{aligned}$$

Boundary conditions, $x = 0 \rightarrow h = h_L$ and $x = L \rightarrow h = h_R$
for $x < \alpha L$

$$\begin{aligned} C_L &= \frac{k_L}{2}h_L^2 \\ C_R &= \frac{k_R}{2}h_R^2 + q_0 L + \frac{1}{2}NL^2 \end{aligned}$$

yielding

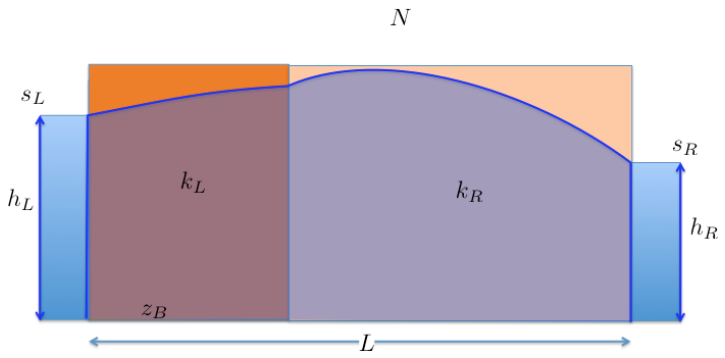

Figure 1: Simple model to use for test calibration, notice h is relative to the bottom of the aquifer, s is relative to an arbitrary fixed datum.

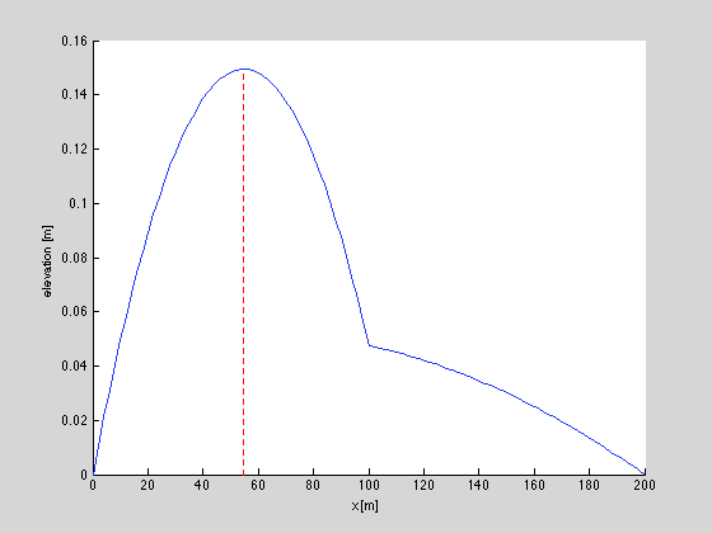

Figure 2: Head in the simple model for default values $z_B = -20$; $s_L = s_R = 0$; $\alpha = 0.5$; $k_L = 1$; $k_R = 20$; $L = 200$; $x = 0 : 2 : L$; $N = 0.002$; All dimensions in m and d units.

$$\begin{aligned}
\frac{k_L}{2}\left(h^2 - h_L^2\right) &= -q_0 x - \frac{1}{2}Nx^2, \quad x \leq \alpha L \\
\frac{k_R}{2}\left(h^2 - h_R^2\right) &= q_0\left(L - x\right) + \frac{1}{2}N\left(L^2 - x^2\right), \quad \rightarrow x \geq \alpha L
\end{aligned}$$

or

$$\begin{aligned}
h^2 &= h_L^2 - \frac{2q_0}{k_L}x - \frac{N}{k_L}x^2, \quad x \leq \alpha L \\
h^2 &= h_R^2 + \frac{2q_0}{k_R}\left(L - x\right) + \frac{N}{k_R}\left(L^2 - x^2\right), \quad \rightarrow x \geq \alpha L
\end{aligned}$$

The maximum head can be found form

$$\begin{aligned}
2\frac{dh}{dx} &= 0 = -\frac{2q_0}{k_L} - \frac{2Nx_{peak}}{k_L} \\
x_{peak} &= -\frac{q_0}{N}, \quad x_{peak} \leq \alpha L
\end{aligned}$$

and

$$\begin{aligned}
0 &= -\frac{2q_0}{k_R} - \frac{2N}{k_R}x_{peak} \\
x_{peak} &= -\frac{q_0}{N}, \quad x_{peak} \geq \alpha L
\end{aligned}$$

Hence, the peak head is always at $-q_0/N$ irrespective of its location being at the left or right of $x = \alpha L$. The maximum head in the cross section follows from

$$h_M = \sqrt{h_L^2 + \frac{q_0^2}{k_L N}}, \quad x \leq \alpha L$$

$$h_M = \sqrt{h_R^2 + \frac{q_0^2}{k_R N} + \frac{2q_0 L}{k_R} + \frac{NL^2}{k_R}}, \quad \rightarrow x \geq \alpha L$$

we may now eliminate q_0 which follows from the heads being the same at $x = \alpha L,\ 0 \leq \alpha \leq 1$.

$$h_a^2 = h_L^2 - \frac{2q_0}{k_L}\alpha L - \frac{N}{k_L}\alpha^2 L^2 \tag{4}$$

$$h_a^2 = h_R^2 + \frac{2q_0}{k_R}(1-\alpha)L + \frac{N}{k_R}\left(1-\alpha^2\right)L^2 \tag{5}$$

$$h_L^2 - \frac{2q_0}{k_L}\alpha L - \frac{N}{k_L}\alpha^2 L^2 = h_R^2 + \frac{2q_0}{k_R}(1-\alpha)L + \frac{N}{k_R}\left(1-\alpha^2\right)L^2$$

$$\begin{aligned} 2q_0 L\left(\frac{(1-\alpha)}{k_R} + \frac{\alpha}{k_L}\right) &= \left(h_L^2 - h_R^2\right) - NL^2\left(\frac{\alpha^2}{k_L} + \frac{1-\alpha^2}{k_R}\right) \\ q_0 &= \frac{\left(h_L^2 - h_R^2\right) - NL^2\left(\frac{\alpha^2}{k_L} + \frac{1-\alpha^2}{k_R}\right)}{2L\left(\frac{\alpha}{k_L} + \frac{(1-\alpha)}{k_R}\right)} \end{aligned} \tag{6}$$

assume $k_R = k_L$ and $\alpha = 0.5$, $h_L = h_R$

$$q_0 = -\frac{NL}{2}$$

which is correct.

With equations 4, 5 and 6 we now have a suitable analytic test model with sufficient degrees of freedom to experiment with calibration. This model has been implemented in the file model.m. It has a selftest that produces the figure above.

Considering that in practice we have the heads and the recharge given, we have now at least 4 free parameters that would be uncertain in reality: k_L, k_R, α and z_B, the elevation of the bottom of the aquifer to adjust in order to make our model fit the data..

We will see to what extent we may calibrate some or all four of them given the heads in the ditches, the recharge and the head measurements.

If we compare this to a real case, our model may be in error because it misses essential ingredients, such as the entry resistance of the ditches. Maybe the aquifer is underlain by another aquifer with which it exchanges water through an aquitard. We might also misinterpret the distribution of the recharge, which may not be equal in practice due to different land use for instance. Of course, the conductivity may not be properly zonable into two distinct zones, perhaps it has more zones or the conductivity varies in a more continuous fashion. Or the jump from one zone to the other is at a location different from what we assumed. We may as well have too many parameters in our model. This is always the case if there are more parameters than there are measurements. But it may also be that the transmissivity is so homogeneous that trying to estimate two conductivities does not make sense. In all cases where the model is a misconception of the real system, we may expect the optimized parameters compensate to the extent possible for the misconceptions in our model, and therefore, are wrong. Although in some circumstances, the model may show a reasonable fit with the date even with erroneous parameters, one may expect it to yield completely useless (uncertain) results when used for predictions under changed circumstances.

4 Use of the model

The model allows passing of a number of parameters. It will use default values for any unspecified parameter. How the input works can most easily be seen in the selftest function inside the model mfile.

The calibration is done with the script Calibration. It shows the four parameters and a usage through which parameters can be switched on and off. The initial parameter values are then specified. This is followed by computation of synthetic measurements, name y_M. These are generated by setting the parameters to the initial ones with some offset, after which random errors are added.

As soon as we have we can compute the sensitivities of the heads at all measurement locations with respect to all available parameters. This is done by computing the heads for the initial parameter set and then in turn for all parameters while their value is changed a bit in turn. Then the initial heads area subtracted and the difference is divided by the parameter change that was used.

Having computed the sensitivities, we can compute the optimal parameters as was outlined above. This is done assuming the computed values at measurement locations varies linearly with the parameter values. We then compute the heads at the measurement locations using the updated parameters. We then have the difference between the optimized model and the measurements. Form this we can compute statistics, such as

The covariance matrix of the parameters.

The correlation matrix of the parameters.

The standard error of the heads.

And of the parameters (assuming the linearity between computed values and parameters.

The uncertainty of the parameters.

In this script we compute only one update of the parameters. In the real world of model calibration, we would start anew with the now updated parameter values. Updates are only necessary because of the non-linearity between model outcomes and parameter values. The parameter values have to be updates into the direction of the global optimum, assuming that such an optimum exists, which is to be made plausible by starting the calibration several times, each time starting with a substantially different initial parameter set. Each time the calibration should end with the same parameter end values. If not the calibration outcome is non-unique, i.e. the model errors are not sensitive to the parameters or there are more parameters than measurements, independent measurements.

It is interesting to see how many measurements are needed for a reasonable parameter optimization. It is also interesting to see how the uncertainty changes with the number of parameters that are included. In general, more parameters give a better fit but at the same time a higher parameter uncertainty.

```
%% Calibration simple analytic model
% TO 130619

clear variables

%% Parameters used in the calibration
usage =     [ 1     1       1     1 ];
parname = {'kL','kR','alpha','zB'};

use = usage~=0;

%% initial parameter values and perturbations
kL    = 1;        dkL = 0.05*kL;
kR    = 20;       dkR = 0.05*kR;
alpha = 0.25;     da  = 0.05*alpha;
zB    = -10;      dzB = 0.05*zB;
L     = 1000;

% Change of default of model parameters, add any (see model for possible parameters
defaults = {'L',L};

%% Number of observations and measurement locatons
Np    = 50;
xM    = unique(L * rand(Np,1));  % random locations between 0 and L

%% initial parameter vectors, usage above determines which ones are used.
```

```
p0  =[kL kR alpha zB]';  % initial paramter vector
dp0 =[dkL dkR da dzB];   % change applied to compute sensitivity (Jacobian)


%% Generate measurements
%
% The measurements are generated using the model with parameters a bit offset
% from the true parameters and with random errors added.
% Offset from true parameters
off_kL    =-0.2*kL;
off_kR    = 0.2*kR;
off_alpha = 0.2*alpha;
off_zB    = 0.0*zB;

% true parameters that will make the model equal to the measurements
pTrue = [kL+off_kL,kR+off_kR,alpha+off_alpha,zB+off_zB]';

% true model without random errors
yM = model('xM',xM,'kL',kL+off_kL,'kR',kR+off_kR,'alpha',alpha+off_alpha,'zB',zB+off_zB,de

try   % try to load random errors (to keep them the same all the time)
    load randErrors
    if numel(yM)~=numel(randErrors)
        error('Generating random errors');
    end
    fprintf('Random errors loaded.\n');
catch ME  % renew random errors
    fprintf('%s\nGenerating and saving random errors.',ME.message);
    randErrors = 0.05*randn(size(yM));
    save randErrors randErrors;
end

% simulated measurements
yM = yM + randErrors;

%% Initial parameters and model outcome
% The initial parameters for the calibration were given above.
% We have at most 4 parameters in this model kL kR alpha and zB.
% The active ones are selected with the usage near the top of this file.

%% Sensitivities computation (Jacobian)

% Model outcome for initial parameters
y0=  model('xM',xM,'kL',kL,'kR',kR,'alpha',alpha,'zB',zB,defaults{:});

%% perturbation of model parameter values
% Run model for all parameters in turn with a small value change
sp= [
    model('xM',xM,'kL',kL+dkL,'kR',kR,'alpha',alpha,'zB',zB,defaults{:}),... % par1
    model('xM',xM,'kL',kL,'kR',kR+dkR,'alpha',alpha,'zB',zB,defaults{:}),... % par2
    model('xM',xM,'kL',kL,'kR',kR,'alpha',alpha+da,'zB',zB,defaults{:}),...  % par3
    model('xM',xM,'kL',kL,'kR',kR,'alpha',alpha,'zB',zB+dzB,defaults{:})     % par4
    ];
```

```
%% Compute Jacobian matrix (sensitivities)
J = bsxfun(@rdivide,bsxfun(@minus,sp(:,use),y0),dp0(use));

%% Optimal update of initial parameters
Inv     = (J'*J)^(-1);
B       = Inv*J';

dp      = B *(yM-y0); % dp  = (J'*J)^(-1)*J' * (yM-y0)

y       = y0+J*dp;     % end results, initial + update through parameter change
p       = p0(use)+dp;  % end results for parameters

%% Show results for comparison
fsz = 14; % fontsize plot

figure; axes('nextplot','add','fontsize',fsz);
xlabel('x [m]','fontsize',fsz);
ylabel('head [m]','fontsize',fsz);
title('Calibration: Head in measurement points','fontsize',fsz);

plot(xM,yM,'bx'); % model measured data
plot(xM,y0,'ro'); % model initial parameters
plot(xM,y ,'gs'); % model optimized parameters
legend('measured','initial','optmized');

%% Covariance matrix and other statistics
e       = (yM-y);                % heads errors, measured - computed
sigma   = std(e);                % errors in heads after calibration
Cov     = sigma^2*Inv;           % covariance matrix of the parameters
sigmaP  = sqrt(diag(Cov));       % std of the parameters
uncert  = 100*sigmaP./abs(p);    % uncertainty
Cor     = Cov./(sigmaP*sigmaP'); % correlation matrix of the parameters

%% Display results
display(Cov);
display(Cor);

%% Issue results for the parameters in readable format
fprintf('results: error = %.4g m\nUncertainty = 100*sigmaP/abs(p)\n',sigma);
fprintf('%10s%10s%10s%10s%10s%10s\n','parameter','pTrue','pInit','pEnd','sigmaP','uncert%'
k=0;
for i=find(use)
    k=k+1;
    fprintf('%10.4s',parname{i});
    fprintf('%10.4g',pTrue(i));
    fprintf('%10.4g',p0(i));
    fprintf('%10.4g',p(k));
    fprintf('%10.4g',sigmaP(k));
    fprintf('%10.4g',uncert(k));
    fprintf('\n');
end

%% Then next step is to change the initial parameters into the correct
% direction
```

5 Weighted least squares and maximum likelihood estimation

The text above descrives least squates optimization, which is generally most applies in regression analysis. Least squares estimaiton yields unbiased results. Nevertheless, it may be necessary to apply weighted least squares estimation, when the value of data varies much or to deal with the contribution of different data types to the cost function that is to be minimized. Different data types like heads, flows and concentrations are generally not compatible in value magnitude, dimension and importance for the model. A flow or flux may add much more to the stability of a calibration than heads. A flux is always necessary to obtain a unique solution. In the test model we chose the recharge to fulfill that role. It generally makes no sense to try to calibrate the recharge. This is because the heads depend on the ratio of the recharge over the aquifer transmissivity and without recharge, transmissivity cannot uniquely be determined.

One issue not discussed above is weighted least squares or maximum likelihood optimization. The latter contains information allowing to tradeoff between more parameters with a better fit and fewer parameters but with lower uncertainty.

6 Exercise

Is is a good exercise to embed this model in a global optimization that will update the parameters until the parameter values have converged to stable end values. This has been done in model2. This can be readily done in Matlab using Marquardt Levenberg optimization function. It may also be done with external programs like UCODE or PEST. HOwever, for instruction purposes, it may be most convenient to do the entire calibration within Matlab. It helps in understanding what these dedicated calibrating methods actually do. But more importantly one should grasp a feeling for the relation between parameters, model fit and uncertainty.

References

[1] Cooley, R.L. and R.L. Naff (1990) Regression Modeling of Ground-Water Flow. Techniques of Water Resources Investigations of the United States Geological Survey. Book 3, Applications of Hydraulics, 228pp.

[2] Doherty, J (2000) The Pest Manual. Model-Independent Parameter Estimation and Uncertainty Analysis. See for a list of literature references http://www.pesthomepage.org/Some_References.php.

[3] Doherty, J (2013) See http://www.pesthomepage.org/Downloads.php There are tutorials and downloads on that site. The software is free and intiernationally intesively used for calibration of models of any type.

[4] Hill, M. and C.R. Tiedeman () Effective Groundwater Model Calibration with Analysis of Data, Sensitivities, Predictions and Uncertainty. Wiley 2007, 13- 978-0-471-77636-9. 455p

[5] Methods and Guidelines for Effective Model Calibration. USGS Water Resources Report 98-4005. With applications to UCODE, a computer code for universal inverse modeling, and MODFLOWP, a computer code for inverse modeling with MODFLOW.

[6] Olsthoorn, T.N (1998) Groundwater modelling: calibration and the use of spreadsheets. PhD thesis. TUDelft, 300pp

[7] Stark, H. andJ.W. Woods () Probability, Random Processes and Estimation Theory for Engineers. Second Edition, Prentical Halll, 1994. ISBN 0-13-728791-7. 617 pp.