Parameter Estimation in Groundwater Engineering

a.k.a. model calibration, site characterization, Inverse modeling, History matching...

Harry Lee

4/11/2018

CEE 696

Table of contents

Parameter Estimation

Introduction : Parameter Estimation/Inverse Problem

- Given perfect description on *HK*, MODFLOW predicts the state of the system, i.e., hydraulic head *h*

Introduction : Parameter Estimation/Inverse Problem

- Given perfect description on HK, MODFLOW predicts the state of the system, i.e., hydraulic head h
- In practice, we measure (noisy) data d, which is a subset of h

Introduction : Parameter Estimation/Inverse Problem

- Given perfect description on *HK*, MODFLOW predicts the state of the system, i.e., hydraulic head *h*
- In practice, we measure (noisy) data *d*, which is a subset of *h*
- Given *d*, we need to estimate *HK*
- This problem is called "site characterization", "model calibration", "model parameter estimation", "inverse problem" or "history matching"
- Let's say n_{obs} is the number of observations and $n_{unknowns}$ is the number of unknown *HK* values

Introduction : Parameter Estimation/Inverse Problem

- Given perfect description on *HK*, MODFLOW predicts the state of the system, i.e., hydraulic head *h*
- In practice, we measure (noisy) data *d*, which is a subset of *h*
- Given *d*, we need to estimate *HK*
- This problem is called “site characterization”, “model calibration”, “model parameter estimation”, “inverse problem” or “history matching”
- Let’s say n_{obs} is the number of observations and $n_{unknowns}$ is the number of unknown *HK* values
- typically $n_{obs} \ll n_{unknowns}$

Introduction : Parameter Estimation/Inverse Problem

- Given perfect description on *HK*, MODFLOW predicts the state of the system, i.e., hydraulic head *h*
- In practice, we measure (noisy) data *d*, which is a subset of *h*
- Given *d*, we need to estimate *HK*
- This problem is called “site characterization”, “model calibration”, “model parameter estimation”, “inverse problem” or “history matching”
- Let’s say n_{obs} is the number of observations and $n_{unknowns}$ is the number of unknown *HK* values
- typically $n_{obs} \ll n_{unknowns}$
- Error in *d* and MODFLOW leads to uncertainty in estimation *HK*

Introduction : Parameter Estimation/Inverse Problem

- Given perfect description on HK, MODFLOW predicts the state of the system, i.e., hydraulic head h
- In practice, we measure (noisy) data d, which is a subset of h
- Given d, we need to estimate HK
- This problem is called “site characterization”, “model calibration”, “model parameter estimation”, “ inverse problem” or “history matching”
- Let’s say n_{obs} is the number of observations and $n_{unknowns}$ is the number of unknown HK values
- typically $n_{obs} \ll n_{unknowns}$
- Error in d and MODFLOW leads to uncertainty in estimation HK
- Therefore, HK should be estimated in a statistical framework

Example

https://github.com/jonghyunharrylee/pyPCGA/blob/master/examples/modflow_flopy/inversion_modflow.ipynb

or you can run this inversion test on our Jupyter server.

Regression Analysis/Model fitting

Intro to Regression analysis

We first start with "overdetermined case", i.e. $n_{obs} > n_{unknowns}$

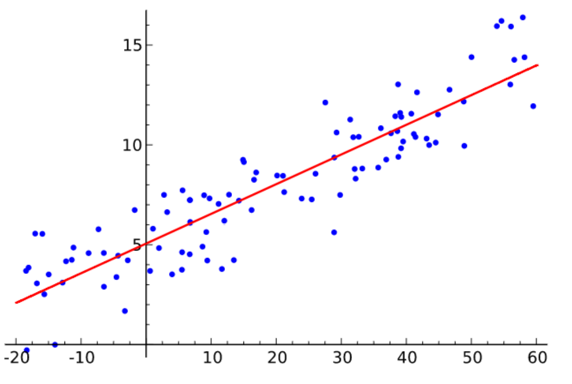

Figure 1: example of regression analysis from Wikipedia
`https://en.wikipedia.org/wiki/Least_squares`

What if $n_{obs} == n_{unknowns}$?

Regression analysis/Model Fitting

Given we have data ($\mathbf{x}$,d), e.g, $\mathbf{x}$ is the location of the hydraulic head/water level measurement d, we would like to set up a mathematical model (whether it be physical, statistical or empirical) to reproduce and predict h given x.

Let assume our model is linear, i.e.,

$$h = ax + b$$

Then if we use “Least Squares” approach, we would like to find a and b that minimizes the sum of squared residuals (i.e. difference, model error and so on)

Connection to optimization

"Least Squares" is most widely used method for this curve fitting:

$$\min_{a,b} \sum (d_i - ax_i - b)^2$$

Now parameter estimation becomes an optimization problem!

Of course, this is not only our minimization criteria. We can minimize their absolute distance/error

$$\min_{a,b} \sum |d_i - ax_i - b|$$

or, maximum error

$$\min_{a,b} \max_{i} |d_i - ax_i - b|$$

and so on. The result can change drastically depending on method and model selection and data (outliers).

scipy.optimize fitting methods

Equation (Local) Minimizers

leastsq(func, x0[, args, Dfun, full_output, ...])	Minimize the sum of squares of a set of equations.
least_squares(fun, x0[, jac, bounds, ...])	Solve a nonlinear least-squares problem with bounds on the variables.
nnls(A, b)	Solve `argmin_x || Ax - b ||_2` for `x>=0`.
lsq_linear(A, b[, bounds, method, tol, ...])	Solve a linear least-squares problem with bounds on the variables.

Global Optimization

basinhopping(func, x0[, niter, T, stepsize, ...])	Find the global minimum of a function using the basin-hopping algorithm
brute(func, ranges[, args, Ns, full_output, ...])	Minimize a function over a given range by brute force.
differential_evolution(func, bounds[, args, ...])	Finds the global minimum of a multivariate function.

Rosenbrock function

rosen(x)	The Rosenbrock function.
rosen_der(x)	The derivative (i.e.
rosen_hess(x)	The Hessian matrix of the Rosenbrock function.
rosen_hess_prod(x, p)	Product of the Hessian matrix of the Rosenbrock function with a vector.

Fitting

curve_fit(f, xdata, ydata[, p0, sigma, ...])	Use non-linear least squares to fit a function, f, to data.

Exercise - curve fitting

Use a simple model (a low-order polynomial, log, exponential function) to fit your research data.

Use scipy.optimize.curve_fit or scipy.optimize.least_square

We will continue with underdetermined case ($n_{obs} < n_{unknowns}$) in the next class