GSTDMB 2010
Dynamical Modelling for Biology and Medicine:

Parameter estimation and Sensitivity analysis

Markus Owen

1 Parameter estimation and Sensitivity analysis

Parameter estimation and *Sensitivity analysis* are very broad topics which are often treated separately, but are in fact quite closely related. In this practical we consider fitting the parameters of two quite simple models. MATLAB functions and other files referred to in this practical can be found in the zip archive at:
http://www.maths.nottingham.ac.uk/mathsforlife/gstdmb2010/prac-parameters.zip

1.1 Yeast growth

In the lectures we discussed trying to fit yeast growth data to simple population models. The data is contained in the above zip archive as "`CEN-PK113-5D.xls`".

We define the error between the data and model solutions by:

$$E = \sum_j \left(x(t_j) - x^{data}(t_j) \right)^2,$$

where t_j is the set of time points at which experimental measurements were taken (column one of the data), $x^{data}(t_j)$ is the yeast optical density at those time points, averaged over the three experimental replicates (columns 2–4 of the data), and $x(t_j)$ are the corresponding values of the model solution. We use the first 1500 minutes of the data.

As in the lectures, given a solution feature ϕ, we define the sensitivity of that feature to changes in the parameter p as

$$S_p^\phi = \frac{\delta\phi/\phi}{\delta p/p},$$

i.e. the fractional change in the feature divided by the fractional change in the parameter. Thus, if a 10% increase in parameter p gives rise to a 10% change in the feature ϕ, the sensitivity is equal to one. Sensitivities of magnitude greater than one correspond to parameter changes which induce a greater proportional change in the solution feature.

We would like to know how sensitive the error is to parameter changes away from a previously identified *optimal* parameter set. Thus, for our purposes, the obvious choice of *feature* is the error itself. If the sensitivity of the error is large, we would expect it to be easier to find the best fit parameter set (since this sensitivity should mean that variations from the optimal set would give quite different solutions). In contrast, if such sensitivities are small, it may be problematic to find a single best-fit, since large changes in parameter values would not give significant changes in the error that we are trying to minimise (another way to think about this is that the error *landscape* is flat).

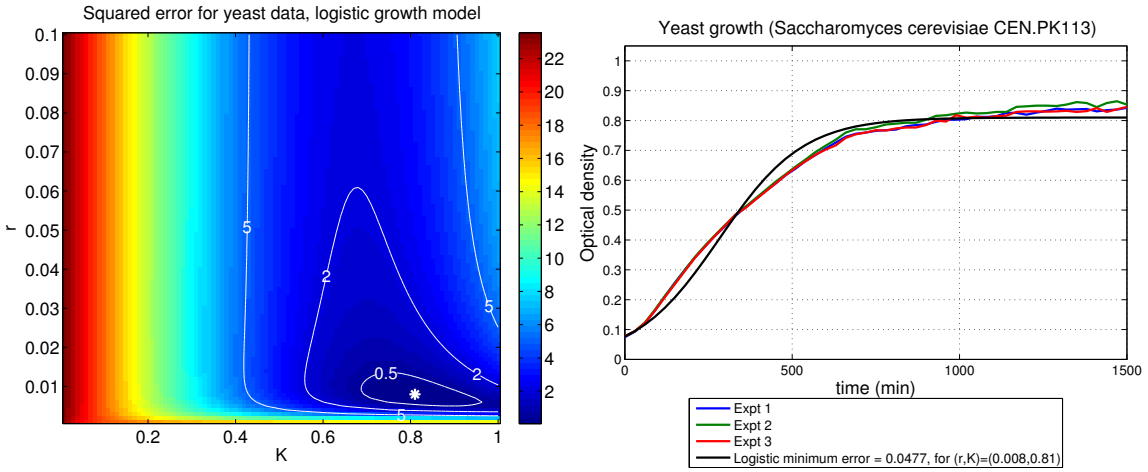
1.2 Fitting the standard logistic growth model

The first step is to consider how good a fit can be achieved with the standard Logistic Growth model:

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{K}\right).$$

This model has only two parameters, r and K (if we treat the initial population size as known), and an analytical solution can be written down, so it is possible to rapidly scan the parameter space to calculate the error between model solutions and average of the three data sets.

- Open and run the MATLAB m-file "`yeastscanlogistic.m`", which contains a function to do this parameter space scan. This function loads the data, then loops over (r, K) space calculating and storing the error in order to generate an image coloured according to the size of the error (dark blue corresponding to the smallest error). Subfunctions are defined at the end of the file for the model solution ("`logsol`") and the error ("`yeastError`"). You should see that there is a clear region of parameter space in which the error is minimised.



The smallest error for parameters included in the scan (which has a predefined resolution of 0.001 in r and 0.01 in K) is found at $(r, K) = (0.008, 0.81)$.

- Open and run the MATLAB m-file "`yeastfitlogistic.m`", which contains a function to carry out a *search* for the best fit parameter set using "`fminsearch`", which implements the Nelder-Mead Simplex algorithm (a local search method). This function finds a more accurate estimate of the optimal parameter set $(r, K) = (0.00760224, K = 0.81619)$, since it is not constrained to a particular resolution of parameter space.

  "`yeastfitlogistic.m`" includes a subfunction "`yeastErrorsens`" which calculates the sensitivity of the error to parameter changes of $\pm 1\%$ and $\pm 10\%$. Look at the output in the MATLAB command window:

```
********************
Running YEASTFITLOGISTIC.M ...
********************
Initial guess: r=0.01, K=0.7.

Logistic model fit to yeast data:  r=0.00759966, K=0.81621, x0=0.0763477. Error=0.0424155
Matrix of sensitivities: top row=fractional parameter change, 1st column=parameter number
         0    0.0100   -0.0100    0.1000   -0.1000
    1.0000    0.6131   -0.6641    5.4544   -7.5567
    2.0000    5.0534   -5.0599   50.2170  -50.9383
```

This shows that the error is most sensitive to changes in parameter 2 (the carrying capacity K). However, it also has a sensitivity of magnitude greater than one to changes of $\pm 10\%$ in r (parameter 1). Given the results of the parameter scan this should not be a surprise—there is a well-defined optimal parameter set (even though the actual fit is not that good).

You might want to edit these lines of the file "`yeastfitlogistic.m`":

```
% ------------------------------------------------------------
% ************************************************************
% CHANGE THESE: INITIAL GUESSES AT MODEL PARAMETERS
% See how the parameters found and their sensitivity changes
% ------------------------------------------------------------
% pars is the set of parameter values, [r,K,v,x0]
% this search needs a reasonable initial guess - here are a couple
guess1 = [0.01,0.7];
guess2 = [0.01,0.8];
guess3 = [0.14,0.9];
% try the routines with one of the guesses
pars = guess1;
% ------------------------------------------------------------
```

to change the initial guess, but you should find that you get almost identical results.

Next we will see that we can get a better fit (smaller error) using the generalised logistic model, but it is not tightly constrained, even though the individual parameter sensitivities are significant.

1.3 Fitting the generalised logistic growth model

The Generalised Logistic Growth model is:

$$\frac{dx}{dt} = rx\left(1 - \left[\frac{x}{K}\right]^v\right)$$

and the solution can be written

$$x(t) = \frac{K}{\left(1 + \left(\left(\frac{K}{x_0}\right)^v - 1\right)e^{-rvt}\right)^{1/v}},$$

where $x0 = x(0)$, the population size at time $t = 0$. Thus there are four parameters to fit: r, K, v and $x0$.

A MATLAB function to fit the generalised logistic growth model to the first 1500 minutes of the data can be found in the m-file "`yeastfit.m`".

The function "`yeastfit.m`" is rather complex. It plots the data, then generates best fits for the above analytical solution. In addition, as in the previous section, "`yeastfit.m`" calculates the sensitivity of the objective function to variations in the parameters away from their optimised values.

At first, you should only edit these lines of the file:

```
% ------------------------------------------------------------
% ************************************************************
% CHANGE THESE: INITIAL GUESSES AT MODEL PARAMETERS
% See how the parameters found and their sensitivity changes
% ------------------------------------------------------------
% pars is the set of parameter values, [r,K,v,x0]
% NOTE: the initial value is a parameter to fit
% for initial guess we could use the mean from the data
```

```
x0 = mean(yeast(1,2:4));
% this search needs a reasonable initial guess - here are a couple
guess1 = [0.1,1,0.1,x0];
guess2 = [0.1,1,1,0.1];
guess3 = [1,1,0.01,0.1];
% try the routines with one of the guesses
pars = guess1;
% ************************************************************
% ------------------------------------------------------------
```

for example, by changing "`pars = guess1;`" to "`pars = guess2;`".

- Run "`yeastfit.m`". How does the error compare to that given by the best fit for the standard logistic model of the previous section?
- What do you notice about the parameter sets that are found for different initial guesses? Do the parameter sets for different initial guesses agree?
- How do the parameter sensitivities vary?
- In the lecture, we saw that for fixed $x0$ and K, the remaining two parameters, r and v are not tightly constrained, but rather that the error is very small for a large range of r and v values. For code to generate that figure, see"`yeastscanfixy0andK.m`".

  The shape of the region with the smallest error suggests that as v increases, r can decrease to compensate, even though the individual parameter sensitivites are relatively high. Recently the notion of *sloppy parameter sensitivities* has been introduced into the literature to try to take account of this kind of joint parameter dependence.
- DIFFICULT: Can you explain why r and v cannot be tightly constrained?

1.4 A two-variable example

Consider the following positive feedback model for mRNA x encoding a transcription factor y:

$$\frac{dx}{dt} = \frac{\lambda y^m}{\theta^m + y^m} - \mu_1 x$$

$$\frac{dy}{dt} = \delta x - \mu_2 y.$$

The file "`feedback.m`" simulates this model, and adds random perturbations to the true solution at selected time points, generating "experimental" data to test our ideas about parameter fitting. The data is also written to a file "`feedback.txt`" which contains three columns of data points—the first column is time, the second and third columns are the x and y data values at those times. Try running "`feedback.m`" as follows:

```
>> data = feedback
Running FEEDBACK.M ... xrand = 0, yrand = 0

data =

         0    0.0500    0.0100
    6.6667    2.4513    0.7432
   13.3333    3.6914    1.3463
   20.0000    4.3282    1.6641

>> allpars = [0.5, 3, log(0.005), 0.1, 0.2, 0.5, 0.05, 0.01];
>> data = feedback(allpars,0.1,0.1,4,1)
Running FEEDBACK.M ... xrand = 0.1, yrand = 0.1
```

```
data =

         0    0.0492    0.0093
    6.6667    2.5593    0.6826
   13.3333    3.3224    1.2619
   20.0000    4.1571    1.6127

>>
```

The second use "`data = feedback(allpars,0.1,0.1,4,1)`" adds noise to the true solution.

The file "`feedbackfit.m`" generates data like this and then tries to find the original model parameters in the same way as the previous example. Open this file and run it in MATLAB. It should generate output as below, which shows the true parameters and those generated by the fitting algorithm:

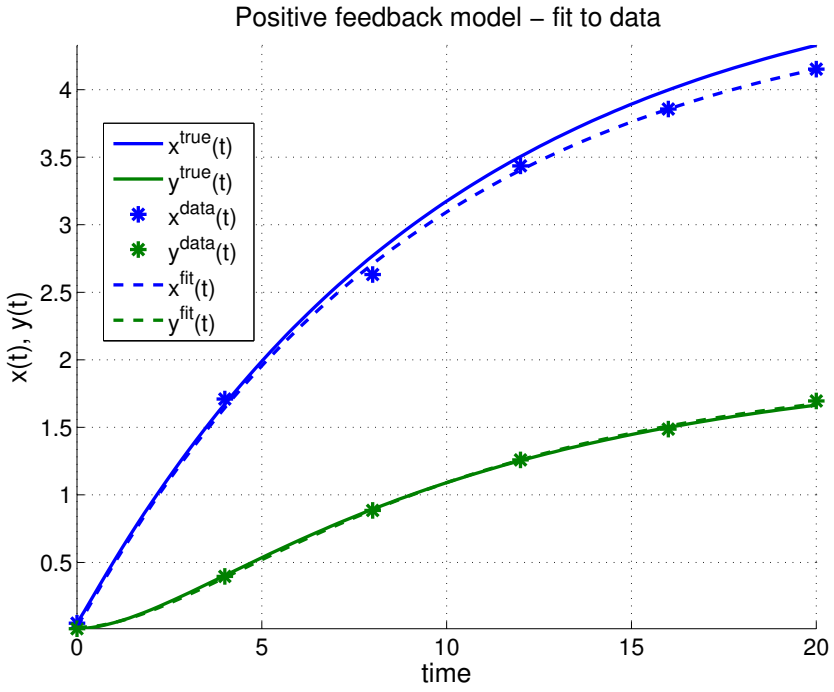
```
>> feedbackfit
********************
Running FEEDBACKFIT.M ...
********************
Running FEEDBACK.M ... xrand = 0.05, yrand = 0.05

lambda          = 0.5, 0.503059
m           = 3, 6.35305
theta           = 0.005, 0.00857058
mu1           = 0.1, 0.106929
delta           = 0.2, 0.18205
mu2           = 0.5, 0.431544
x0          = 0.05, 0.0123319
y0          = 0.01, 0.0211893

Finished FEEDBACKFIT.M ...

>>
```

and a plot like the one below which shows that the fit to the generated data is good (and that the fit is close to the true solution). However, looking at the parameter values we see that some are quite far from their "true" values.



Try manipulating the parameters of the model and the initial guess for optimisation, to see what effect these have. As before, you should only edit these lines of the file (unless you really feel like going further):

```
% ------------------------------------------------------------
% ************************************************************
% CHANGE THESE: TRUE SYSTEM/MODEL PARAMETERS
% See how the changing the parameters affects the fitting process
% ------------------------------------------------------------
% set the model parameters to generate "experimental" data
% the set of parameter values, [lambda,m,theta,mu1,delta,mu2,x0,y0]
lambda = 0.5;
m = 3;
theta = 0.005;
mu1 = 0.1;
delta = 0.2;
mu2 = 0.5;
% the initial conditions are also paramaters
x0 = 0.05;
y0 = 0.01;
% noise in x
xrand = 0.05;
% noise in y
yrand = 0.05;
% number of time points
numsamples = 4;
% which parameters to fit - the rest are assumed known
ifitpars = [1 2 3 4 5 6 7 8];
% ifitpars = [1   3   5   7  ];
% this search needs a reasonable initial guess - here are a couple
allguess1 = [0.5,2,log(0.005),0.5,2,0.1,0.005,0.01];
allguess2 = [0.5,3,log(0.005),0.5,2,0.1,0.005,0.01];
allguess3 = [0.5,3,log(0.15),0.5,2,0.1,0.005,0.01];
% try the routine with one of the guesses (select those that are not fixed)
fitpars = allguess1(ifitpars);
% ************************************************************
% ------------------------------------------------------------
```

- What happens if you change the number of sample points (change "`numsamples`")?
- What about different initial guesses (change "`fitpars = ...`")?
- What is the effect of having less (or more) noise?
- What if you only had data for x? You can investigate this by changing the definition of the objective function "`feedbackError`" to omit the contribution of y.
- Do you have any idea why some of the parameters (such as m) are hard to constrain?
- You can fix some parameters and only choose to fit a subset, by changing "`ifitpars`". For example, "`ifitpars = [1 3 5 7 ];`" sets the parameters to fit as λ, θ, δ, and x_0. The remaining parameters are set to their true values. How much does this help?

NOTE: this work should illustrate that parameter estimation is not straightforward and there are many problems and pitfalls.