Parameter Estimation in Magnolia

Magnolia provides the ability to estimate model parameters from observed data using both deterministic methods (i.e., least-squares and likelihood maximization) and Bayesian methods (via MCMC sampling of a user-specified join probability distribution). While the latter method provides a flexible and robust framework for incorporation of prior information regarding parameter values (i.e., statistics) and attribution of parameter uncertainty and variability to various sources (e.g., through the development of hierarchical statistics models), conventional deterministic parameter estimation methods offer a simple and computationally efficient way of obtaining parameter values from observed data.

The implementation of deterministic parameter estimation in Magnolia via the CMD scripting language's FIT command is described here. The "Mathematical Background" section presents a brief summary of the two estimation methods supported in Magnolia, and serves as necessary context to the description of the command syntax and options in the subsequent "Parameter Estimation Using the CMD Scripting Language" section. A few examples are then presented, illustrative of how Magnolia's DATA, SET and FIT commands are used in coordination to perform parameter estimation.

Mathematical Background

The parameter estimation methods implemented in Magnolia's FIT command all rely on numerical maximization or minimization of an objective function, which quantifies how closely the model predictions match a set of corresponding observations. Computing the objective function typically involves running the model (potentially many times), finding the values of model outputs at the times for which observations were recorded, and summing up individual contributions to the objective function at each of these time points. For the least squares method, each contribution to the objective function is determined by the squared residual: the difference between the observed value and the predicted value at that time. An optional weight can be applied to each residual so that the relative contributions of large and small predicted/observed values can be controlled. The "optimal" set of parameter values, then, is obtained by maximizing this objective function. For the maximum likelihood method, the objective function represents the overall probability of observing a particular set of outputs (the observed data), given the predictions made by the model using a particular set of parameter values. The overall probability is given by the product of the individual probabilities associated with each observed/predicted value pair. If an assumption is made that observed values are normally distributed around some "true" mean (corresponding to the value predicted by the model), then each contribution to the likelihood is the value of the normal distribution at the observed value, with a mean determined by the predicted value along with an appropriate value for the standard deviation (more on this below). In this case, the optimal set of parameter values would correspond to the value that maximize this objective function. In practice, it's computationally more practical to compute the logarithm of the objective

function (the log-likelihood function), meaning the product over sample points is replaced by a sum.

In both cases, computing the inputs to the objective function requires running the model. Moreover, finding the minimum/maximum of the function over the possible values of the parameters is an iterative process. Thus, depending on the number of parameters to be estimated, and the amount of data used for fitting model parameters, a single parameter estimation analysis may require running the model many times. A brief review of the numerical optimization algorithms used in Magnolia is provided in the "Numerical Optimization" section below.

Least Squares (LS) Estimation

Let θ represent the collection of parameters to be estimated. Let $\hat{y}_{i,j}$ represent the value of model output *i* at timepoint *j* (i.e., the *predicted* value). Similarly, let $y_{i,j}$ represent the value of an observed quantity corresponding to the model output with index *i* at timepoint *j*. The (unweighted) sum of squares objective function is then defined by:

$$f_{LS}(\theta) = \sum_{i,j} \left(\hat{y}_{i,j}(\theta) - y_{i,j}\right)^2$$

where the index *i* ranges over the number of observable quantities (model outputs) and the index *j* ranges over the number of observations for each output. The argument θ makes explicit the fact that the values of the predictions, and thus the objective function, depend on specific values of the parameters used when calculating the objective function.

Normally, observed data includes a certain amount of variability or measurement error. That error may be predominantly absolute (i.e., independent of the "true" value), relative (i.e., proportional to the "true" value), or something in between. Thus, in some cases, computing the objective function as shown above may cause large observed values to have disproportionate influence on the sum compared to smaller ones. This situation can be addressed by assigning a weight to each contribution to the sum:

$$f_{LS}(\theta) = \sum_{i,j} w_{i,j}\left(\hat{y}_{i,j}(\theta) - y_{i,j}\right)^2$$

where the summation indices are the same as in the previous equation. The individual weight values are given by:

$$w_{i,j} = \frac{1}{\left|\hat{y}_{i,j}(\theta)\right|^{2\gamma}}$$

where γ is a user-specified value in the range [0, 1]. A value close to zero indicates predominantly absolute error, while values close to one indicated predominantly relative error. A value in the middle of this range indicates a mixture of absolute and relative error.

Identification of optimal parameter values thus amounts to optimizing $f_{LS}(\theta)$ subject to any parameter constraints.

Maximum Likelihood (ML) Estimation

As above, let θ represent the collection of parameters to be estimated, and let $\hat{y}_{i,j}$ and $y_{i,j}$ represent the predicted and observed values the model output with index *i* at timepoint *j*. Assuming observed values are normally distributed around a "true" (mean) value corresponding to the model prediction, the probability of observing a particular data point, given a particular set of parameter values, is given by:

$$P_{i,j}\left(y_{i,j}, \theta\right) = N(y_{i,j}, \hat{y}_{i,j}(\theta), \sigma)$$

Here, $N(x, \mu, \sigma)$ represents the normal (Gaussian) probability distribution with mean μ and standard deviation σ, given by:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The standard deviation σ used in this distribution is determined as part of the optimization process, based on a user-specified "error model" setting. The error model indicates how the standard deviation relates to the predicted mean μ ("true" value) based upon an assumption as to whether the error is predominantly additive (i.e., independent of the predicted value), proportional (i.e., proportional to the predicted value), or mixed (a combination of both). Mathematically, the standard deviation may then be expressed in three ways:

Additive Error Model	$\sigma_{i,j} = \eta_{add,i}$
Proportional Error Model	$\sigma_{i,j} = \eta_{prop,i} \cdot \hat{y}_{i,j}$
Mixed Error Model	$\sigma_{i,j} = \eta_{add,i} + \eta_{prop,i} \cdot \hat{y}_{i,j}$

Here, as above, the subscript *i* denotes the index of a particular model output, and the subscript *j* denotes the index of the timepoint at which a particular observation was taken. The quantities $\eta_{add,i}$ and $\eta_{prop,i}$ represent the additive and proportional error, respectively, for the output with index *i*. The values of these quantities are determined as part of the optimization process as if they were actual model parameters.

Note that Magnolia currently requires that the same error model be used for all observed quantities in a model, although the specific $\eta_{add,i}$ and $\eta_{prop,i}$ values are defined on a per-output

basis and optimized individually. Thus, when performing a parameter estimation using a proportional error with two output (observed) quantities, there will be two additional values determined as part of the fitting process, namely, the proportional error for each of the two outputs.

The "likelihood" function (LF) captures the total probability of observing all the samples in the entire data set used for estimation, and is found by multiplying the individual probabilities:

$$LF(\theta) = \prod_{i,j} N(y_{i,j}, \hat{y}_{i,j}(\theta), \sigma)$$

In contrast to the objective function for the least squares method, we wish to *maximize* the likelihood function. Maximizing this function, however, is equivalent to maximizing its natural logarithm. Redefining the objective functions as the log-likelihood (LLF), we take the log of the right side of the equation, turning the product into a summation:

$$LLF(\theta) = \sum_{i,j} \log\left(N(y_{i,j}, \hat{y}_{i,j}(\theta), \sigma)\right)$$

Similarly to the case of LS parameter estimation, identification of optimal parameter values for ML thus amounts to optimizing $LLF(\theta)$ subject to any parameter constraints, with the important distinction that we're now maximizing the function instead of minimizing it. In practice, however, the same numerical optimization methods can be applied to both problems, since maximizing any objective function is equivalent to minimizing its negation.

Numerical Optimization

Numerical optimization methods generally consist of iterative algorithms for "searching" a space of parameters to find a set of parameter values (i.e., a point in the parameter space) which minimize (or maximize) some function of those parameters to within a particular tolerance. That tolerance may be defined by criteria such as change in objective function value from one iteration to the next (a vanishing change indicates the solver has "converged" on a particular set of parameters), gradient of the objective function (a vanishingly small gradient indicates an extremum in the objective function), or a vanishingly small change in parameter values from one iteration to the next (also an indication of convergence).

Different optimization algorithms may be more or less suited to a particular application based on a number of criteria:

- The requirement to compute the gradient of the objective function: gradient-based methods generally converge faster, but do so at the expense of having to perform a numerical gradient computation.

- Sensitivity to local minima/maxima in the objective function: deterministic algorithms have the drawback of potentially becoming trapped in local minima/maxima and not finding the desired (global) optimum. For simple problems with a sufficiently well-behaved objective function, this isn't typically a problem. But with problems involving many observables and data sets, some experimentation using different algorithms and different starting estimates of parameters may be useful.
- Ability to handle parameter constraints: when parameter constraints (upper and lower bounds of feasibility, in this case) are available, the parameter search space can be substantially reduced and the chance of successfully finding an optimal set of parameters increased. Constraints are discussed in further detail below.
- Sensitivity to solver settings: most optimization algorithms have a number of adjustable settings, which affect the ability of the solver to find optimum parameter values for differing objective functions. Solvers which require substantial amounts of hand-tuning these settings for each problem can be somewhat difficult to use in practice.

Magnolia supports three optimization algorithms, as follows:

- **Nelder-Mead (downhill simplex method)**. The Nelder-Mead algorithm uses a heuristic to directly search the parameter space for an extremum in the objective function. While simple to implement and generally robust, the Powell and BOBYQA algorithms (described below) tend to perform better. In cases where the Powell or BOBYQA methods fail to converge, however, it's occasionally useful to use Nelder-Mead first to find an approximate optimum, then use Powell or BOBYQA to refine the estimates. The Nelder-Mead algorithm is described in Nelder and Mead (1965).
- **Powell (Powell's conjugate direction method)**. This algorithm finds an optimum of the objective function by iteratively performing a bidirectional line search over a set of direction vectors (using Brent's method). This set of vectors, as well as the current point in parameter space, is updated on each iteration, without the need for derivative calculations. This algorithm is described in Powell (1964).
- **BOBYQA (Bound Optimization BY Quadratic Approximation)**. BOBYQA is a derivative-free optimization method described in Powell (2009), which is well-suited for problems involving a large number of parameters. BOBYQA is the default algorithm used by the FIT command, and generally out-performs the Powell method.

Constrained vs Unconstrained Optimization

Frequently, it's possible to identify an upper or lower limit on the range of feasible parameter values (e.g., a parameter that's only physically meaningful for positive values). In this case, providing information on these limits in the form of parameter constraints can speed up convergence of the parameter estimation and help prevent the solver from getting stuck in local (undesired) minima/maxima or venturing into an area of the parameter space that produces numerical instabilities in the solution of the ODEs.

Different optimization algorithms handle constraints in a variety of ways. Currently, Magnolia supports simple upper/lower bound constraints as described above when using the BOBYQA algorithm. For the Nelder-Mead or Powell methods, constraint information is ignored. It is possible to effectively constrain values of parameters to be positive even when using Nelder-Mead or Powell, however, by using log transformation of parameter values (described below).

Use of upper/lower bounds on parameters can also serve a diagnostic purpose. If, at the conclusion of the parameter estimation run, one or more of the final parameter values coincides with an upper or lower bound, the results should be regarded with suspicion. In this case, the optimizer likely did not find a true optimum in the parameter space, and this the parameter value may not represent an optimal fit of the model to the observed data.

Scaling of Parameters

When multiple model parameters are being estimated simultaneously, optimization convergence issues can arise when the scales of those parameters differ by orders of magnitude. In these cases, applying scaling of parameter values can facilitate convergence of the optimization algorithm. Magnolia supplies two types of parameter scaling transformations for use in parameter estimation, as, described below.

Linear Scaling

When upper and lower bounds are provided for parameters, it's possible to scale the parameters so that, from the optimizer's point of view, all parameters are in the range [0, 1]. Specifically, the supplied upper and lower bounds are used to perform a linear transformation of the parameters so that the effective parameter space is a unit hypercube. As the optimizer evaluates the objective function at various points in the space, the inverse transformation is applied to the (transformed) values of the parameters before they are used by the model. This kind of parameter "normalization" is useful when appropriate bounds can be identified for the parameters, or for cases in which parameters may have negative values.

Log Transformation

Frequently, model parameters are known to be positive (i.e., only positive values correspond to physically meaningful situations), but little else is known about their relative ranges. In this case, log transformation of parameters has the advantage of scaling parameter values so that their values are prevented from being negative while compressing ranges of parameter values which may differ by many orders of magnitude in the untransformed parameter space. This type of transformation is common when estimating, for example, kinetic parameters in chemical and biological systems.

Parameter Estimation Using the CMD Scripting Language

Parameter estimation in Magnolia requires coordinated use of the DATA, SET and FIT commands within a CMD language script. Observed data must be specified in comma-separated-value (CSV) file format.

Specifying Observed Data Using the DATA Command

The DATA command tells Magnolia what CSV files contain the data to be used for parameter estimation, how columns in those files map to the values of simulated time or model outputs corresponding to observed quantities, and how prescribed model parameters (i.e., parameters other than the ones being estimated) capture the conditions specific to each experimental data set (i.e., factors or treatments).

Format of CSV Files Used by the DATA Command

Generally, each column in the data file represents a time series of observed values. Each DATA command contains a list of mappings of columns to model outputs. Consider the following segment of a CSV file named "obs_01.csv':

```
Time,  A1,    A2,    B1,    B2
0.0,   1.32,  0.0,   0.84,  0.53
1.1,   1.12,  0.13,  0.68,  0.38
2.0,   0.84,  0.36,  0.31,  0.20
3.3,   0.52,  0.79,  0.19,  0.13
```

A column header is required, but the specific names of each column are arbitrary. Columns are all mapped to time and model outputs using the syntax of the DATA command:

```
DATA @file='obs_01.csv' ds1 t='Time' a='A1' b='B1' tstop=4 c=1.23
```

The @file flag is used to specify the name of the data file; note that multiple files may be used as inputs to a single parameter estimation run. The "ds1" argument provides a unique name for this particular dataset, used predominantly to create plots from DATA commands and to specify data sets in the context of Bayesian parameter estimation (described elsewhere). The name is arbitrary, but must be unique among the data sets used for a specific estimation run. The next three arguments are where Magnolia is told how to map data file columns to model variables. The identifier on the left side of each equals sign denote the name of a model variable, or t (the independent variable). In this example, "t" is the model variable representing time and "a" and "b" are other model variables (outputs). The name enclosed in single quotes on the right side of each equals sign specifies which column in the CSV file provides observed values for that output. In this example, "A1" corresponds to observed values for model variable "a" and "B1" provides observed values for model variable "b". Again, note that the identified

on the left side of the equals sign mush be a valid model variable name, but the names of the columns in the CSV file are arbitrary. The “tstop” and “c” arguments to this command relate to experimental conditions/factors, as described below.

The manner in which files encompass data from multiple experiments is somewhat flexible. It’s possible to construct a parameter estimation script in which all data resides in a single file, or in which data is spread out over multiple files. Magnolia loosely regards an “experiment” as a set of data corresponding to a specific set of experimental conditions, where the experimental conditions (sometimes called “descriptor” variables/parameters) are represented by one or more prescribed model parameter values. For example, in a chemical kinetics experiment, the parameter being estimated may be a kinetic rate parameter, while the experimental conditions (descriptors) for each data set might be the initial concentrations of the reactants or catalysts.

Returning to the example above, the following two DATA commands could be used to tell Magnolia that data from two different experiments should be used to estimate model parameters:

```
DATA @file='obs_01.csv' ds1 t='Time' a='A1' b='B1' tstop=4 c=1.23
DATA @file='obs_01.csv' ds2 t='Time' a='A2' b='B2' tstop=4 c=2.54
```

The first command indicates that the data for the first experiment (for which the prescribed value of “c” is 1.23) is contained in columns “A1” and “B1, ” where “A1” contains the data corresponding to model variable “a,” and “B1” contains the data corresponding to model variable “b”. Likewise, the second command indicates that observations for a second experiment comes from the save data file, but with the observed values for “a” coming from column “A2” and the observed values for “b” coming from column “B2”. For the second experiment, the factor “c” had a value of 2.54.

Alternatively, if the data was captured in two separate data files, the equivalent specification of input data may have looked something like this:

```
DATA @file='obs_01.csv' ds1 t='Time' a='A' b='B' tstop=4 c=1.23
DATA @file='obs_02.csv' ds2 t='Time' a='A' b='B' tstop=4 c=2.54
```

In this case, each file contained a single column for observed values of model outputs “a” and “b,” but the way in which the data is used by Magnolia to estimation parameter values is entirely equivalent to the first example, and will produce the same estimates.

Note that a single period (“.”) may be used to indicate a missing value in a data set. Missing values are simply skipped when computing contributions to the objective function as described in the “Mathematical Background” section above. Note also that the values of time corresponding to observed samples need not align exactly with the communication interval of the model (i.e., the time step at which model outputs are logged, denoted in the model code by the CINTERVAL statement). Magnolia controls simulation stepping so that outputs are

computed at the exact observations times specified in the CSV file, irrespective of the value of communication interval.

Setting Parameter Initial Estimates and Constraints using the SET Command

As mentioned in the "Mathematical Background" section, the optimization algorithms used to perform parameter estimation in Magnolia are iterative; that is, they gradually produce optimal values for parameters by improving upon a previous set of value through various methods of traversing the parameter space. Thus, the algorithms need an initial set of estimates for the parameters to start the process. The initial "guesses" for the parameters are specified using the SET command, where the value used by the solver is simply the value specified in the last SET command before the FIT command is issued.

The SET command has optional @min and @max flags that can be used to specify upper and lower bound constraints on the parameter search space, as described in the "Mathematical Background" section. Note that the bounds used in the last SET command before the FIT command is issue are the one that will be used by the solver. Also note that simple constraints are only used when the BOBYQA optimization algorithm is selected (this is the default algorithm used in Magnolia). So, to specify the initial value of a model parameter named "d" to 5.2, and to specify that the parameter search should be limited to the interval [0.1, 100], the following command could be used:

```
SET d=5.3 @min=0.1 @max=100
```

Additional SET commands would be specified for any additional parameters to be estimated.

Estimating Parameters using the FIT Command

Initiating the actual parameter estimation run is done using the FIT command. This command specifies the set of parameters to be fitted and the settings related to the choice of estimation method, optimization algorithm and error model (maximum likelihood) or residual weighting (least squares). Any specification of data files to be used for parameter estimation, as well as fitted parameter initial estimates and upper/lower bound should have been previously indicated using appropriate DATA or SET commands. Note that DATA commands are effectively applied to any FIT commands within a CMD script. If performing parameter estimation with alternative data sets included or excluded needs to be performed, it's best to set these up in different CMD script files.

The example FIT command below would instruct Magnolia to start a parameter estimation run to estimate the values of model parameters "d" and "e" (whose initial values and bounds would have been specified in previous SET statements), using a maximum likelihood estimation method with a fixed error model:

```
FIT @method=ml @errormodel=mixed d e
```

Details regarding the use of the @method and @errormodel flags, as well as other flags to the FIT command, are presented in the following sections.

Note that, at present, only scalar (i.e., not array) model parameters may be used by the FIT command.

Summary of DATA, SET and FIT Command Syntax and Options

DATA Command

DATA commands are specified using the following syntax:

DATA @file='<filename>' <dataset name> <variable>='<col name>' … <parm name>=<value> …

Arguments/flags for this command are as follows:

@file='<filename>'	'<'filename>' is used to specify the (single-quoted) name of the CSL file. If not path information is included in the file name, the file is assumed by Magnolia to reside in the same folder at the CMD script which contains this DATA command.
<dataset name>	An arbitrary (unquoted) name used to uniquely identify the dataset corresponding to this DATA command within the script. This name should only contain letters, numbers and underscores (i.e., no spaces or special characters).
<variable>='<col name>' …	One or more arguments used to specify how columns in the data file map to model variables (outputs). Each argument consists of a variable name / column name pair. The variable name (unquoted) on the left side of the equals sign should be the name of a model variable (case insensitive). The column name (single-quoted) should be the name of one of the columns (as contained in the column header row) in the corresponding CSV file. Note that the column names are somewhat arbitrary (they don't have to be the same as the model variable name), that all columns in the file don't necessarily have to be used in a particular DATA statement; in fact, multiple DATA statements can refer to the same CSV file by mapping different columns to the same model output.
<parm name>=<value> …	One or more arguments used to specify the value of descriptor variables (factors) capturing experimental conditions specific to each data set. The (unquoted) <parm name> argument on the left side of the equals sign should be the (case insensitive) name of a model parameter (as defined in a CONSTANT statement). The <value> argument on the right side of the equals sign should be a numeric value specifying the value of the experimental condition (factor) corresponding to this data set.

SET Command

SET commands used for parameter estimation are specified using the following syntax:

SET <parm name>=<value> @min=<min value> @max=<max value>

Arguments/flags for this command are as follows:

<parm name>=<value>	<parm name> is used to specify the (unquoted, case insensitive) name of the model parameter for which this SET command provides and initial value. The <value> specified on the right side of the equals sign should be a valid numeric value.
@min=<min value>	The @min flag is used to specify an optional lower bound constraint on the parameter. <min value> must be a valid numeric value.
@max=<max value>	The @max flag is used to specify an optional upper bound constraint on the parameter. <max value> must be a valid numeric value.

Note that the SET command also supports a number of additional flags/arguments which can be used for Monte-Carlo (variability) analysis. But the arguments presented in the table above are the only ones relevant to parameter estimation using the FIT command, and the use of other arguments within CMD scripts specifying parameter estimation run may lead to unexpected results.

FIT Command

The arguments of the FIT command differ depending on whether least squares or maximum likelihood is chosen as the estimation method:

FIT @method=ls @gamma=value <parm name 1> <parm name 2> …
FIT @method=ml @errormodel='<errmod name>' <parm name 1> <parm name 2> …

The FIT command also supports optional arguments (@scaleparms, @logparms, @algorithm) for specifying whether parameter should be scaled and what algorithm should be used to perform optimization of the objective function (as described in the "Mathematical Background" section).

Arguments/flags for this command are as follows:

@method=<meth name>	The @method flag is used to indicate the parameter estimation method to be used. Valid (unquoted) values are @method=ml for maximum likelihood and @method=ls for least squares.
@gamma=<value>	When using the least squares method (i.e., @method=ls), the @gamma flag can be used to specify the value of gamma for weighting residuals, as described in the "Mathematical Background" section.
@errormodel='<errmodel>'	When using maximum likelihood method (i.e., @method=ml), the @errormodel flag can be used to specify the error model used when computing the value of sigma to be used in the standard deviation for each likelihood contribution, as described in the "Mathematical Background" section. Valid (single-quoted) values for this flag for @errormodel='additive' for additive error,

	@errormodel='proportional' for proportional error, or @errormodel='mixed for mixed error.
@scaleparms	Inclusion of this flag causes all parameter values to be scaled to an interval of [0, 1] using the upper and lower bound information included in the corresponding SET command. If both @scaleparms and @logparms are specified, the @logparms flag is ignored. If both @scaleparms and @logparms are omitted, scaling is not performed.
@logparms	Inclusion of this flag causes all parameter values to be log-transformed. Log-transformed parameters must be greater than zero, so zero or negative initial values of these parameters result in an error. If both @scaleparms and @logparms are specified, the @logparms flag is ignored. If both @scaleparms and @logparms are omitted, scaling is not performed.
@algorithm='<algname>'	The @algorithm flag is used to set the optimization algorithm to be used when performing parameter estimation, as described in the "Mathematical Background" section. Valid (single-quoted) values for this flag are @algorithm='bobyqa' to use the BOBYQA algorithm, @algorithm= 'neldermead' to use the Nelder-Mead algorithm, or @algorithm= 'powell' to use the Powell algorithm. BOBYQA is used by default, if no @algorithm flag is specified. Note that BOBYQA must be used is parameter upper/lower bound constraints are to be used in the parameter estimation.
<parm name> …	One or more <parm name> arguments are used in the FIT command to specify which parameters should be fitted by the FIT command. These should all be valid model parameters specified in the CSL code using CONSTANT statements, and corresponding SET commands in the CMD script should generally be placed before the FIT command is invoked.

Examples

Single Observable, Single Experiment

For this example, consider a chemical reaction in which a single reactant A is irreversibly converted to a product B with first-order reaction rate constant k1. This system can be represented by the following model in Magnolia (let's say this is contained in a file named "reaction1.csl"):

```
model reaction1
derivative

    constant k1 = 1, A0 = 1, B0 = 0
    dA = -k1*A
    dB = k1*A

    A = integ(dA, A0)
    B = integ(dB, B0)

    constant tstop = 10.0
```

```
        termt(t >= tstop, 'Stopped on time limit')

    end
    end
```

The variables A and B represent the concentrations of reactant and product, respectively, and constant k1 represents the reaction rate constant. A0 and B0 represent the initial concentrations of each of the chemicals at T=0 (an experimental condition). We would like to estimate this rate constant using observed concentration samples at various times. For this reaction, we only need data on the disappearance of chemical A to do this.

Assume we've performed an experiment with an initial concentration of A equal to 2.0, and no product yet formed (i.e., the initial concentration of B is zero). A CSV file containing the observed data for A might look something like this (let's say the name of the file is "conc_vals_1.csv":

```
Time,  Conc
0.5,   1.39
1,     0.951
1.5,   0.792
2,     0.579
2.5,   0.457
3,     0.265
3.5,   0.226
4,     0.131
4.5,   0.142
5,     0.111
6,     0.0335
```

To construct the script to estimate k1, we need to specify the following:

- The name of the file containing the observed value of A
- The columns in this file that map to model variables T and A
- The initial concentrations of A and B used in this experiment
- The initial value of k1
- Any additional parameter estimation setting (e.g., method, algorithm, etc.)

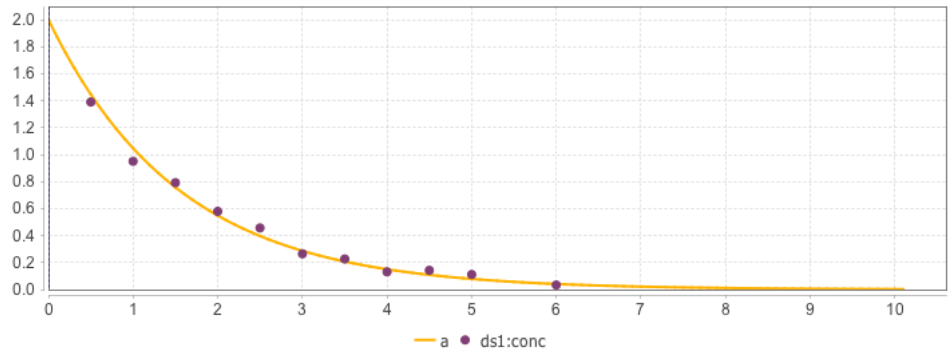
A CMD file which provides all this information would then be:

```
load 'reaction1.csl'

data @file='conc_vals_1.csv' ds1 t='T' a='A' a0=2 b0=0

set k1 = 1

fit @algorithm='neldermead' @logparms k1

plot A 'ds1:A'
```

The first line, containing the LOAD command, is used to specify the model for which we're estimating parameters. Note that this file must be open in Magnolia when the parameter estimation is started. The DATA command specifies the name of the CSV file containing the data (note that this file does NOT need to be open). Because no path information is included in the file name, the file is assumed to exist in the same folder as the CMD script. The DATA command also assigns the name "ds1" to this data set; this name will be used in a subsequent PLOT command to overlay the observations on a plot of the predicted concentration profile of A. The model variables T and A are assigned to columns named "Time" and "Conc" (respectively) in the CSV file using the next two arguments in the command. Finally, the initial values of A and B (experimental conditions) are associated with this data set in the last two arguments to the DATA command. Keep in mind that we're doing this because we might have multiple data sets corresponding to different experimental conditions. The SET command is then used to indicate that the initial value of k1 to be used by the optimizer should be 1. The parameter estimation run is then started using the FIT command. Default values are used for the FIT command with the exception of the @logparms flag, which indicates that the search space for k1 is log-transformed; this has the benefit of constraining values of k1 to be positive. Finally, the PLOT command displays the predicted trajectory of A using the optimized value of k1, overlaid with the observed data.

Running the above in Magnolia will produce the following output in the command window, along with the plot pictured below.

```
Optimization complete.
Num obj. fun. evals = 45
Objective function optimized with value of 0.021670802101367834
at parameter values:
  k1 = 0.6470428300904311

Optimization took 88 ms.
```



Multiple Observables, Single Experiment

Now consider extending the previous example to represent a reversible reaction. To do this, we need to add a second rate constant for the reverse reaction; we'll call this k1. The CSL model for this system then becomes ("reaction2.csl"):

```
model reaction2
derivative

    constant k1 = 1, k2 = 0.5, A0 = 1, B0 = 0
    dA = -k1*A + k2*B
    dB = -k2*B + k1*A

    A = integ(dA, A0)
    B = integ(dB, B0)

    constant tstop = 6.0
    termt(t >= tstop, 'Stopped on time limit')

end
end
```

Similar to the first example, the data for an experiment which samples the concentrations A and B at various time points might look like this ("conc_vals_2.csv"):

```
Time,  ConcA,  ConcB
0.5,   1.49,   0.443
1,     1.24,   0.826
1.5,   1.08,   0.97
2,     0.853,  1.11
2.5,   0.777,  1.22
3,     0.826,  1.18
3.5,   0.709,  1.29
4,     0.74,   1.25
4.5,   0.707,  1.25
5,     0.728,  1.23
6,     0.686,  1.27
```

We'll need to make the following adjustments to the CMD file:

- Load the revised model, which includes the reverse reaction ("reaction2.csl")
- Load the revised CSV data file ("conc_vals_2.csv")
- Set an initial value for the newly added parameter k2; we'll also specify upper and lower bound constraints this time
- Update the FIT command to estimate the k2 as well and k1; we'll also use the maximum likelihood method with a proportional error model this time
- Update the PLOT command to display trajectories for both A and B

After doing this, the revised CMD file will look something like the following:

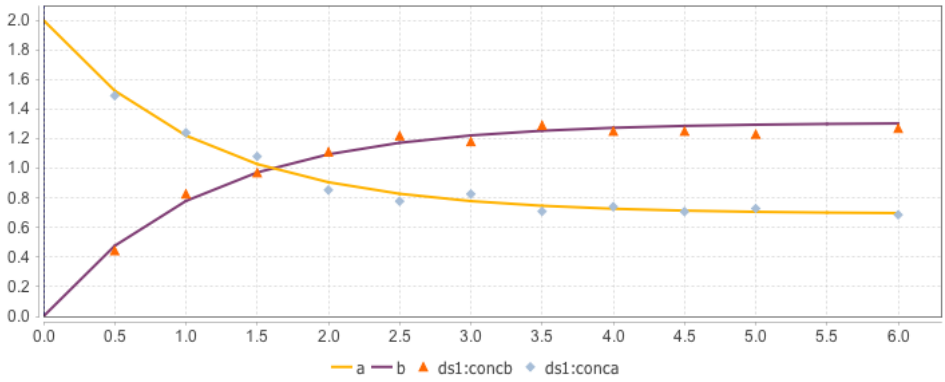
```
load 'reaction2.csl'
```

```
data @file='conc_vals_2.csv' ds1 t='Time' a='ConcA' b='ConcB' a0=2 b0=0

set k1 = 1 @min=0 @max=100
set k2 = 1 @min=0 @max=100

fit @method=ml @errormodel=proportional k1 k2

plot A 'ds1:ConcA' B 'ds1:ConcB'
```

Running the updated CMD script will generate the following output in the command window:

```
Optimization complete.
Num restarts = 1
Num obj. fun. evals = 430
Objective function optimized with value of 29.565784267596065
at parameter values:
  k1 = 0.5920381698329665
  k2 = 0.3128718126732713

Proportional errors:
a: 0.1
b: 0.1

Optimization took 1308 ms.
```

This time, the output displays not only the fitted values of the parameters, but also the optimized values of the proportional error coefficient (as described in the Mathematical Background section) for each model output. The plot now also shows the model fit to the observed values for B:



Multiple Observables, Multiple Experiments

Finally, consider the same reversible reaction of the last example, but imagine that we've performed a second experiment in which the system was initialized with only chemical B at a concentration of 2 at T = 0. So in this case, the model has not changed, but we have additional data to inform our estimates of k1 and k2. The additional CSV file ("conc_vals_3.csv") might be the following:

```
Time,  ConcA,  ConcB
0.5,   0.277,  1.61
1,     0.383,  1.5
1.5,   0.505,  1.39
2,     0.618,  1.43
2.5,   0.707,  1.26
3,     0.669,  1.24
3.5,   0.688,  1.31
4,     0.739,  1.35
4.5,   0.694,  1.19
5,     0.614,  1.28
6,     0.671,  1.24
```

Note that we could have alternatively put the data for the second experiment into the same file as the first, but with new column names. The DATA command in the CMD script would then need to be adjusted accordingly, but the fitted values of k1 and k2 should ultimately be the same. Besides the additional DATA command, we'll refine the upper bounds on k1 and k2 a little to speed up the calculation, and we'll modify the PLOT commands by running the simulation once prior to each PLOT command so that the values of A0 and B0 will be set appropriately for each experiment. The updates CMD script should will look like this:

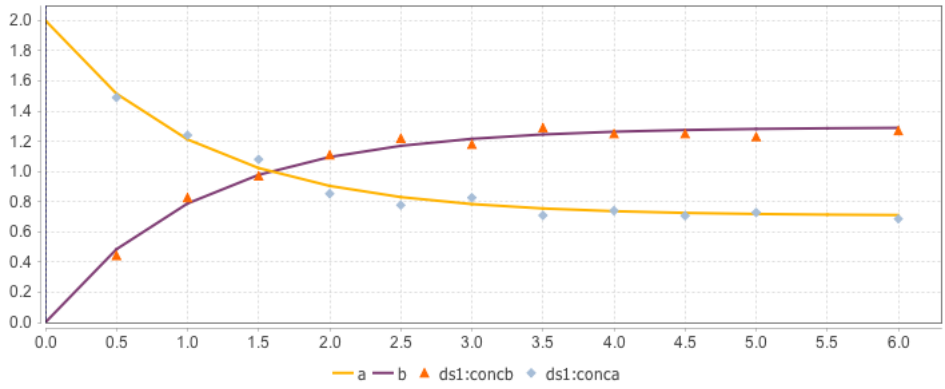
```
load 'reaction2.csl'

data @file='conc_vals_2.csv' ds1 t='Time' a='ConcA' b='ConcB' a0=2 b0=0
data @file='conc_vals_3.csv' ds2 t='Time' a='ConcA' b='ConcB' a0=0 b0=2

set k1 = 1 @min=0 @max=10
set k2 = 1 @min=0 @max=10

fit @method=ml @errormodel=proportional k1 k2

set A0=2 B0=0
start
plot A 'ds1:ConcA' B 'ds1:ConcB'

set A0=0 B0=2
start
plot A 'ds2:ConcA' B 'ds2:ConcB'
```

Running this script will generate the following output:

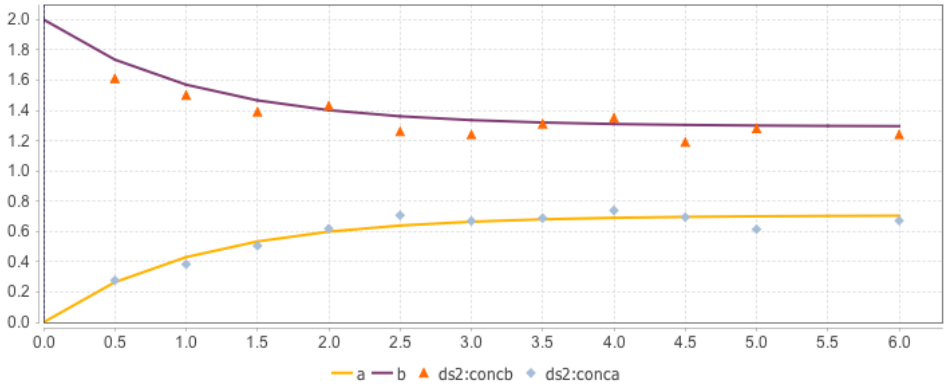
```
Optimization complete.
Num restarts = 1
Num obj. fun. evals = 286
Objective function optimized with value of 58.04788999475382
at parameter values:
  k1 = 0.6076131881312411
  k2 = 0.33197358658599874

Proportional errors:
a: 0.1
b: 0.1
```

```
Optimization took 1303 ms.
```

This time, two plots will be generated: one for each experiment. For the first experiment (A0 = 2, B0 = 0) you should see the following plot:



And for the second experiment (A0 = 0, B0 = 2) you should see the following plot:



References

Nelder, J. A. and Mead, R. (1965). "A simplex method for function minimization". Computer Journal. 7 (4): 308–313.

Powell, M. J. D. (1964). "An efficient method for finding the minimum of a function of several variables without calculating derivatives". Computer Journal. 7 (2): 155–162.

Powell, M. J. D. (2009). “The BOBYQA algorithm for bound constrained optimization without derivatives” (Report). Department of Applied Mathematics and Theoretical Physics, Cambridge University. DAMTP 2009/NA06.