Manual for DEBtoxM

Applications of Dynamic Energy Budget theory in ecotoxicology and stress ecology

Tjalling Jager

tjalling@debtox.info; www.debtox.info
DEBtox Research, The Netherlands

May 9, 2015

Contents

Preface

About this manual

This document provides a manual for my Matlab files to analyse toxicity data. The Matlab files and this document can be obtained from the web site:

http://www.debtox.info/debtoxm.php

The DEBtoxM Matlab files come in several flavours, which are stand-alone collections of scripts and functions. At this moment, I have four flavours:

1. **Surv**: the simplest survival analysis possible, if you have a constant exposure concentration, and don't care about receptors or damage. Choose between stochastic death [1] or a threshold distribution (log-normal). The model is solved analytically, which means that it is lightning fast. This is a good place to start to get to know how my programming works, and to play around with the analysis options.

2. **GUTS**: extensive survival-only analysis, based on the General Unified Threshold model for Survival [2]. Survival only, but a whole range of possible mechanisms can be explored. The model allows for stochastic death, threshold distribution, or a combination of both. Furthermore, a one-compartment damage model, or saturating receptor kinetics can be included.

3. **DEBtox**: new simplified 'DEBtox' derivation (a paper is in preparation, but it is comparable to [9]). Analyse effects on survival, growth and reproduction simultaneously in a simple mechanistic manner, using easy-to-interpret model parameters.

4. **DEB3**: toxicity analysis based on the full standard model in DEB3 [10, 8], as published by [7]. Enormous flexibility, but difficult to use and to interpret correctly. I am not using this version myself anymore, and therefore I decided not to offer it anymore on my web page.

In the future, more flavours might follow, such as the analysis of mixture toxicity. The basic setup of these collections follows the same principles, so all versions can use this single manual. I will explain when a certain feature is specific for a specific flavour.

The DEBtox and DEB3 flavours require background knowledge of DEB theory. As a starting point, I would advice reading my free e-book which can be obtained from:

http://www.debtox.info/book.php

Furthermore, I suggest reading some papers of mine, such as [4, 3, 7, 5].

Support

These flavours are meant to be a research tool, and have not been extensively tested in this form yet. It is therefore likely that you will run into problems or unexpected errors. I do not run a help desk, but you can email me or use the DEBtox board to post your problem:

http://board.debtox.info

Feel free to experiment with model adaptations etcetera, and if you have a nice addition, please let everybody know at the discussion board!

Although an attempt is made to provide a 'user-friendly' environment, the proper use of these scripts requires (basic) knowledge of Matlab, and takes time to get used to (also because I have implemented a lot of flexibility and many 'useful options'). Play with the examples to get a feel for the possibilities. To get started with Matlab, consult the Mathworks web pages:

http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

Alternatives

As an alternative to the DEBtoxM files, I suggest you take a look at my BYOM framework. This is a related set of Matlab files, but it is much more flexible to experimentation and currently updated more frequently. The DEBtoxM files are better suited to analysis with multiple datasets that share common parameters. These files can be downloaded from:

http://www.debtox.info/byom.php

Disclaimer

These files are presented 'as is', without guarantees; I am not a commercial software engineer. I do not accept any liability for errors in the program, or misuse of the program, or consequences that may arise from these situations. However, if you come across an error, or if you have other comments, please use the discussion board, and I might address these issues in future versions.

Please note that the DEBtoxM files are currently 'in a state of limbo'. Most of my current activities are one-off analyses, for which the BYOM framework is better suited. I might return to DEBtoxM at some point, and do some major overhauling, but that may take a while.

Chapter 1

Getting started

1.1 Setting up DEBtoxM

Extract the DEBtoxM files to your computer (the scripts and functions should work equally well under UNIX, Windows, or Apple). The upper-level directory has a name that refers to which 'flavour' of DEBtoxM you are using. You can change this name into any name of your liking; you can have as many of these directories as you like. Please do not change the names of the directories of the levels below. In the directory `/DEBtoxM`, you will find 4 directories. Select the directory `/mydata_examples` as your active directory in Matlab. This directory contains `mydata_*.m` files. A mydata file contains a dataset, specifies what kind of analysis is to be performed, and provides initial estimates for the parameters. In most cases, this is the only file you will need to adapt for your own work. Furthermore, this directory contains a file called `pathdefine.m`. This is called from the mydata file, and checks whether the other subdirectories `/common` (common files for all flavours), `/engine` (specific files for the flavour) and `/utils` (some handy utilities, also common to all flavours) are in the path; if not, they are added (non-permanently).

Setting the path can also be done the hard way using `<set path>` from the file menu, or the `path` command in Matlab. If you have multiple sets of DEBtoxM collections or flavours (like I do), this is not so practical as you would have to reset your path every time you switch between collections. Working with `pathdefine.m` is easier, but take care: the path is set the first time you run a mydata file. If you want to switch to a different collection, make sure to remove the DEBtoxM directories from the path, or exit Matlab, restart Matlab, and go the new collection. That way, you are sure that the path points to the correct locations.

The mydata files are the starting point for each analysis. You can have as many mydata files as you like, so make a new one for every data set that you want to analyse and give it an appropriate name. You can make multiple directories and sub-directories with mydata files, but make sure they are all somewhere below `\DEBtoxM` (at the level of `\DEBtoxM` you could for example put your original Excel data files). This provides the opportunity to organise your work in the way you like, e.g., with sub-directories for different species or different toxicants. Each directory with mydata files should contain a copy of the file `pathdefine.m`. Make sure you keep an unchanged `mydata_empty.m` file somewhere safe for reference purposes. Basically, all analyses can be run by only changing the mydata files. The scripts and functions should work with standard Matlab, and do not require (as far as I remember) any additional toolboxes. The only exception is the optional calculation of posterior parameter distributions with the slice sampler (part of the statistics toolbox).

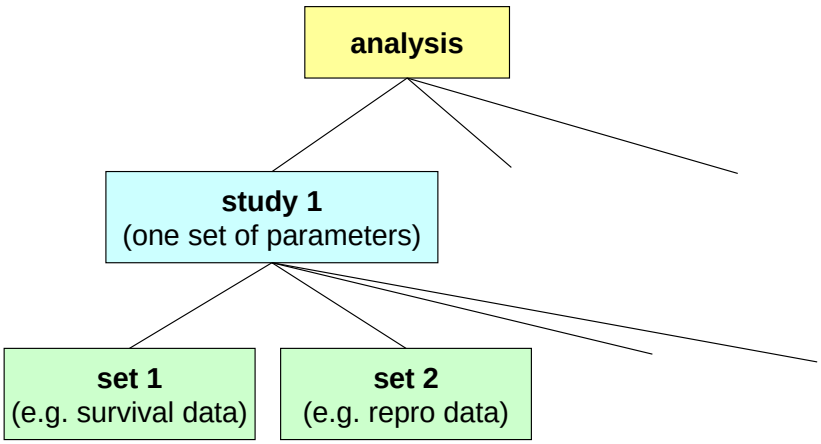

Figure 1.1: Basic structure of the data within an analysis in DEBtoxM.

1.2 Data structure in the mydata file

The philosophy behind DEBtoxM is to maximise flexibility, and to analyse multiple data sets simultaneously (as many as you like). An analysis with DEBtoxM contains one or more 'studies'. A study, in this context, is a number of datasets that share all parameter values (different studies thus differ in at least one parameter). Each study contains one or more 'sets', which hold the actual observations for an endpoint (e.g., survival or reproduction). In principle, there is no limit to the number of studies within a single analysis, or the number or type of sets within a single study (although estimating a large number of parameters from a large amount of data is usually very time consuming, and requires good starting values).

The analysis level

```
TYPES.name = 'Fake data set'; % identification of overall analysis, for reporting only
% Concentrations and times may contain duplicate values, and can be in any order.
% However, the concentration and time vectors must match the observation matrix.

TYPES.units = {'mg' 'L' 'd' 'mm' 'kg'}; % units for chemical mass, environmental volume/weight,
% time, body length, body weight for output
% Note that you can choose any units you like, as long as you use it consistently throughout!
```

At the analysis level, you can specify an overall name for the analysis. This name will appear in the output on screen, but also in the title of the figures that are produced. Furthermore, you have to specify the units for chemical mass, environmental volume/weight, time, body length and body weight (you need to provide five text strings here, even if you do not have e.g., body weights). These units will also be used in the output (there is no check on consistency of units, or anything that fancy; it is for helping interpretation of the output). Both the analysis name and the units are strings, which is why they are placed between quotes in the code.

The study level

```
study = 1;
ADD_tmp(study).studyname = 'Study 1'; % specify what all the individual studies are, for reporting only
```

At the study level, you need to provide a number. Number the studies starting with 1, call the next one 2, etcetera. This sounds trivial, but numbering them non-sequentially creates problems when running the files. Further, each study gets a name (again as a string) to specify what makes this study different from the others. This name is only used in the output (both on screen and in the figures).

The set level

```
% internal concentration
sett = 1;
type = 1; % body residue data
T = [0 10 20]'; % time vector of data
C = [0 1 3]'; % concentration vector of data

% Enter body residue data as matrix (time in rows, conc in columns)
N = [0 0 0
     0 0.4 2
     0 1.2 3.5];

% the weight factor should be the number of individuals per data point
W = [10 10 10
     10 10 10
     10 10 10];
```

Each set within a study gets a number (numbering starts from 1 in each study, and again must be sequential). Note that the variable that contains the number is called `sett`, with double t's, because `set` is a built-in Matlab command. Each set requires specification of a type; what endpoint is this data for (in this example, the type is 1, which means internal concentrations). The following types are allowed:

type	endpoint	comment
0	external conc.	fitted or splined, only GUTS flavour, one set per study
1	internal conc.	
2	survival data	as survivors or as time-to-death
3	body length data	only DEBtox/DEB3 flavours, as length or weight$^{1/3}$
4	reproduction data	only DEBtox/DEB3 flavours, as offspring per time interval

Next, the time vector of the observations, and the concentration vector with the exposure concentrations are placed in the vectors `T` and `C` (as column vectors). The observations on the endpoint are entered as a matrix in the variable `N`. Each column represents the behaviour in time for one exposure concentration. The matrix `N` is linked to the vectors `T` and `C`; it must have the same number of columns as the length of `C`, and the same number of rows as the length of `T`. The values in the time and concentration vector do not need to be unique; if you have, for example, two controls (a solvent and a true blank) you can use two zeros in the concentration vector, and use two columns for the corresponding observations in `N`. Missing data can be included as a `NaN` ('not a number') in the data matrix. For reproduction, anything happening after the first missing data point will not be plotted because reproduction is plotted cumulatively (the rest of the data points *are* used for the likelihood function though). The missing data point is thus interpreted as a point where the offspring is removed without counting (they are *not* expected to be included in the next observation interval).

Next, a weight factor can be provided for each number in the observation matrix `N`. Thus, the matrix `W` must have the same size as `N`. If the observations in `N` are from individual animals, it makes sense to create a matrix `W` with ones. If the observations in `N` are averages from a number of individuals, the weight factors should represent the number of individuals responsible for each average. Thus, averages that are based on fewer animals receive a less weight in the optimisation. The weight factors can also be used to increase or decrease the importance of a specific observation in the data set (e.g., because a data point is considered an 'outlier'). Weight factors have no effect for external concentration sets (which are not fitted; discussed in more detail later).

For survival, the matrix `W` is used in a different way; weighing is not generally useful for survival data (which are the counts for the number of animals alive at each point). Instead, the `W` matrix is used to provide information on animals that were removed from the test unit alive (e.g., for analysis of body residues) or gone missing (e.g., escaping from the test unit). The matrix `N` should contain the uncorrected data, i.e., the number of observed animals in the test container at each census day. The matrix `W` should be the same size as `N`, and contains the number of animals removed alive (or last seen alive) at each census day. In this way, the information that they were alive up to a certain time point is included in the model fit. When the data matrix `N` contains one or more `NaN` values, there should not be anything else than a zero in the matrix `W` at the corresponding positions. Otherwise, the missing/removed animals at those time points are ignored, and a warning is produced by the consistency check on the data set.

In the figures, the data are plotted as fraction survival, corrected for the missing or removed animals. An uncorrected remaining fraction is plotted as a dotted line to judge the impact of the missing or removed animals. If this extra line makes your plot messy, you can turn this off by setting the variable `missplot` in the top of the file `plot_all.m`. If no animals were missing or removed in the test, you can simply use all zeros for `W`:

```
W = zeros(size(N));
```

For non-survival data, when we have a data set with individual measurements on multiple animals, we can fill the observation matrix `N` with the means or with the individual replicates. If we use the means, weight matrix `W` should be filled with the number of replicates for each mean. If we use the individual data, every column of `N` represents an individual observation at a certain exposure concentration. The concentration vector `C` contains duplicate exposure concentrations, and `W` should be filled with ones. Perhaps an example will help here!

Suppose we have a dataset with two replicates per concentration, two exposure concentrations, and measurements at three time points. One of the replicates dies before the end of the studies (use a `NaN` in the observation matrix).

```
T = [0 10 20]'; % time vector of data
C = [0 0 1 1]'; % concentration vector of data

N = [0.1 0.2 12 9
     0.2 0.4 25 31
     0.3 0.2 41 NaN ]; % NaN means not a number

W = [1 1 1 1
     1 1 1 1
     1 1 1 0];
```

If we used the averages of the observations at each exposure concentration, we would obtain the following set:

```
T = [0 10 20]'; % time vector of data
C = [0 1]'; % concentration vector of data

N = [0.15 10.5
     0.30 28
     0.25 4];

W = [2 2
     2 2
     2 1];
```

In general, using the individual replicates is preferable, as such a data set would contain more information.

Different sets do not need to have the same concentration or time vectors, and you can have more sets of the same type in one study (e.g., when you have replicated the experiment). An exception is for external concentrations, for now only allowed in the GUTS flavour, where only one set is allowed. The external concentrations are either fitted assuming a first-order decay, or splined: a continuous line is drawn through the data points, and the exposure concentration interpolated at each time point that is needed in the ODE solver. This is especially useful when the exposure in the test was pulsed. As an example, consider a chemical that is degraded in the exposure medium, and the organisms are transferred to clean water at $t = 20$:

```
ADD_tmp(study).fitext = 0; % fit external concentrations (1) or spline them (0)
type = 0; % external concentration data
T = [0 10 20 20.1 30]'; % time vector of data
C = [1 3]'; % concentration vector of data
% (only used as name for the treatment when external conc. data are provided)

% Enter external concentration data as matrix (time in rows, conc in columns)
N = [1.1 3.2
     0.8 2.7
     0.6 2.5
     0   0
     0   0];

W = ones(size(N));
```

In this piece of code, two exposure patterns are specified. The matrix `N` now contains the measured (or estimated) exposure concentrations at three time points (0, 10 and 20), for the two scenarios. The other time points ($t = 20.1$ and $t = 30$) are zero, which represents the transfer to clean medium. The values in the concentration vector `C` are now only used as names for the two exposure scenarios (in this case, they could have represented the initial nominal concentrations). The model calculations for the (scaled) internal concentration will use the interpolated value from the matrix `N` at each time point. External concentration are not fitted with a model, so the weight matrix is trivial. It still needs to be filled, so here it is just a matrix with ones of the same size as `N`.

For survival data (`type` 2), enter the number of survivors (animals alive) at each time point. Do *not* correct for missing/removed animals; use the weight matrix `W` for that. Alternatively, data can be entered as time-to-death, using the function `ttd2surv` to calculate it to the correct survivor matrix, time vector and weight matrix (for missing/removed animals). See the comments in the `mydata_empty` file for the correct way to call this function, and the

correct way to deal with missing/removed animals. For body size data (`type` 3), enter the data as either actual body length, or the cubic root of weight or volume (which is a sort of length measure). For reproduction data (`type` 4), enter the data as the observed number of eggs or juveniles at the observation time point (not cumulatively, not per female, but the total production in a test unit). This is thus the number of offspring produced between this observation time point and the previous one. The observation intervals do not need to have the same duration. As an example:

```
type = 4; % repro data
T = [0 5 10 20]'; % time vector of data
C = [0 1 3]'; % concentration vector of data
% enter repro data as no of offspring per group of females observed at a time point (time in rows, conc in columns)
% make sure the first time point has zero reproduction!
N = [0 0 0
     0 0 0
     300 250 100
     1100 950 500];

% the weight factor should be the number of individuals per data point
W = [10 10 10
     10 10 10
     10 10 10
     10 10 9];
```

Here, we have three exposure concentrations, each with 10 females. At the observation times $t = 0$ and $t = 5$, no offspring were observed. At $t = 10$, 300 offspring were found in the control. These 300 were produced by the 10 females in this replicate in between $t = 5$ and $t = 10$; an average reproduction of 30 offspring per female in this interval. In the highest concentration ($c = 3$), one female has died in the last interval (at $t = 20$, only 9 females are left). We do not know exactly when this female has died, so we cannot know how much she participated in the production of those 500 offspring. As a pragmatic choice, I take the average: the 500 offspring have been produced by 9.5 females; an average of 52.6 offspring per female in this interval.

In case we are sure that the reproduction data only represent the animals that were alive during the entire interval, the first row of the matrix `W` should be filled with -1. This signals to the likelihood calculation and plotting functions to only work with the number of females at the end of the interval. This situation occurs in one data set that I have. The animals were kept individually, and the experimenter decided to ignore the last count of eggs produced by females when they were dead or missing at the end of an interval.

The last part of the set specifications contains these lines:

```
lam = 0.5; % lambda, for transformation of the data (0=log, 0.5=square root, 1=none)
DATA_tmp(study,sett) = fill_struct(T,C,N,W,type,lam); % collect everything in DATA_tmp cell array
```

The variable `lam` is used to specify a data transformation. Transformation can help to bring the residuals (the difference between model and data) in closer correspondence to a normal distribution (which is assumed in the optimisation for all endpoints apart from survival). For external concentrations and survival data, the value for `lam` has no effect. A second value in `lam` (making it a 1 by 2 vector) can be used to specify a scatter variance, which will be explained later.

In case you use individual measurements instead of means, the data point that is plotted will be the mean on the selected transformation scale. So if `lam = 0`, the geometric mean

will be plotted (with the total range). For reproduction data, a transformation (so `lam` of less than 1) can have strange consequences for calculation of these means. Reproduction data are plotted as cumulative over time, and especially when we have individuals that do not reproduce in an interval, this can lead to a mean cumulative reproduction that is below the actually observed range. The range is calculated from the individual cumulated values; cumulating the mean after transformation could lead to lower values. I therefore advice caution when using a transformation on reproduction data, especially when using observations on individuals.

In the last line, the information specified in the set is collected in the overall data set `DATA_tmp`. It is 'temporary' as you can later on decide to analyse only a part of the data set (treated in the next section). In running the script, there will first be a consistency check of this complete data set. This check will spot the most obvious errors, such as when the size of the observation matrix `N` does not match the time vector `T`.

1.3 Selecting parts of the data set to analyse

Often, it is useful to study a data set in parts; for example, starting with one endpoint only, or only the control data, removing time points where bias is expected. You do not have to make a new `mydata` file for each of these cases; the script allows to make selection:

```
study_select = [1 3]; % which study/studies to calculate?
set_tot = [1 2 4]; % which sets in each study to analyse?
TYPES.nstudy = length(study_select); % number of studies to analyse
for study = 1:TYPES.nstudy,
    ADD(study).studyname = ADD_tmp(study_select(study)).studyname; % select the correct study names
    for sets = 1:length(set_tot),
        T = DATA_tmp(study_select(study),set_tot(sets)).T; % withdraw data from structure
        C = DATA_tmp(study_select(study),set_tot(sets)).C;
        N = DATA_tmp(study_select(study),set_tot(sets)).N;
        W = DATA_tmp(study_select(study),set_tot(sets)).W;
        type = DATA_tmp(study_select(study),set_tot(sets)).type;
        lam = DATA_tmp(study_select(study),set_tot(sets)).lam;
        cl = length(C); % number of concentrations
        % ind_c = [1]; % select concentrations to be included in dataset, e.g. only control
        ind_c = 1:cl;% all concentrations are included in dataset
        ind_t = find(T<+inf); % modify exposure times included in the analysis
        DATA(study,sets) = fill_struct(T(ind_t),C(ind_c),N(ind_t,ind_c),W(ind_t,ind_c),type,lam); % put data back
    end
end
```

You can specify which of the studies you want to analyse, out of all studies that you have entered in the full data set. You can also select specific sets to use in each study. In this case, I have specified that only study 1 and 3 are to be analysed, and only sets 1, 2 and 4 in each study. It is also possible to make a different selection from the sets in each study, but that would require a little reworking of the script, e.g.,:

```
study_select = [1 3]; % which study/studies to calculate?
TYPES.nstudy = length(study_select); % number of studies to analyse
for study = 1:TYPES.nstudy,
    if study_select(study) == 1,
        set_tot = [1 2];
    elseif study_select(study) == 3,
        set_tot = [3 4];
    end
```

in which case, set 1 and 2 are used from study 1, and set 3 and 4 from study 3.

The nested `for` loops in the script run through all studies and sets that you want to analyse, and extracts the data again. Now, you make a selection for concentrations and time points by manipulating the variables `ind_c` and `ind_t`, respectively. In this case, the script will use all concentrations and all time points:

```
cl = length(C); % number of concentrations
% ind_c = [1]; % select concentrations to be included in dataset, e.g. only control
ind_c = 1:cl;% all concentrations are included in dataset
ind_t = find(T<+inf); % modify exposure times included in the analysis
```

However, if I want to use only concentration 1 and 4, and only the time points less than 10 days (or whatever the time unit in the data set is), I would change these lines to:

```
cl = length(C); % number of concentrations
ind_c = [1 4]; % select concentrations to be included in dataset, e.g. only control
% ind_c = 1:cl;% all concentrations are included in dataset
ind_t = find(T<10); % modify exposure times included in the analysis
```

Now, this selection is made for all studies, and all sets. You can modify this to your liking by working with `if` statements, selecting on the study or set. For example, to make this selection only for set 4 in study 1:

```
cl = length(C); % number of concentrations
if study_select(study) == 1 && set_tot(sets) == 4,
    ind_c = [1 4]; % select concentrations to be included in dataset, e.g. only control
    % ind_c = 1:cl;% all concentrations are included in dataset
    ind_t = find(T<10); % modify exposure times included in the analysis
else
    ind_c = 1:cl;% all concentrations are included in dataset
    ind_t = find(T<+inf); % modify exposure times included in the analysis
end
```

In the last part of the selection procedure, the selected part of the data set is placed into the analysis dataset `DATA` (this is the data set that is actually used for fitting and plotting):

```
DATA(study,sets) = fill_struct(T(ind_t),C(ind_c),N(ind_t,ind_c),W(ind_t,ind_c),type,lam); % ...
```

1.4 Defining the type of analysis and post-analysis to perform

The next choice to make in the mydata file is to specify the type of analysis to perform. For this purpose, several 'switches' can be set in the script. These switches are different for the different DEBtoxM flavours. However, several are common, such as the selection of the dose metric (the toxicokinetic variable linked to the toxic effect) and the nature of the scatter structure:

```
% ========================================================================
% Select dose metric to be used (M), TYPES.dosemetric
% ========================================================================
%         1 : external concentration
%         2 : scaled internal concentration
%         3 : actual internal concentration
%
% ========================================================================
% How to derive the scatter variance, TYPES.scattervar
% ========================================================================
% 1) every set has a different error variance
% 2) all sets with the same type within a study have the same error variance
% 3) all sets with the same type have the same error variance
% 4) use supplied error variance (after transf.) as second argument in lam for each set
...
% Switches for analysis types
TYPES.dosemetric         = 3; % select the dose metric to be used (see table above)
TYPES.scattervar         = 1; % how to select the error variance (default = 1)
```

The `TYPES.dosemetric` allows you to select the dose metric (M in GUTS terminology). All flavours support the use of external, internal and scaled internal concentration as the dose metric. The GUTS flavour additionally sports damage and receptor kinetics as option. Based on which dose metric you select, some parameters will have no effect in the analysis (and will not be reported). For example, if you use the scaled internal concentration, the bioconcentration factor P_{Vd} will have no effect on the survival data. When the dose-metric is 3 (actual internal concentration), the NEC (z in GUTS terminology) does not have the interpretation of an external concentration without effects, but is an internal threshold. Also, when you use saturating intake of the chemical into the body, the NEC looses this interpretation. In these situations, I make a recalculation of the (median) external effects threshold and report it below the parameter estimates in the output.

The `TYPES.scattervar` lets you select the assumption for the scatter structure for graded endpoints (body residue, body size and reproduction). Options 1-3 treat the variance of the error as a nuisance parameter that can be different for each set (1), is the same for all sets with the same type within one study (2), or is the same for all sets with the same type over all studies (3). Option 4 allows to use a given variance for each set. This variance needs to be entered as a second number in the parameter `lam` for each study (which thus becomes a 1 by 2 vector). If you choose to transform the data, the variance should be given *after* the transformation.

Other common switches are the ones for the selection of standard errors and/or confidence intervals, and the selection of plots to make:

```
% Switches for standard errors and confidence intervals
TYPES.ProfPars           = 0; % parameters that will be profiled vector with [parnr study]
                              % e.g. [pospar.ke 1; pospar.b 1]
TYPES.ASEswitch          = 0; % calculate asymptotic standard errors? (1 is 'on') (procedure not very robust!)
TYPES.POSTswitch         = 0; % calculate the posterior distribution with MCMC and uniform priors
                              % (slow!) (enter number of samples, 0 is 'off')
% Switches for specific plots
TYPES.PlotoverSwitch     = 2; % 1) plot figures from different studies on top of each other (0 is 'off')
                              % 2) plot the different exposure scenarios in separate figures (esp. for pulses)
TYPES.PlotdoserespSwitch = 0; % also plot dose-response for last time-step, as well as iso-effect lines
                              % (not when there are external concentrations!!)
```

The `TYPES.ProfPars` can be used to make profile likelihoods for selected parameters. Parameters that are to be profiled are indicated by their name (that is also used for initial

parameter values further in the mydata file). The number following the parameter name indicates the study for which the parameter is to be profiled (remember that studies can differ in their parameter values). If you want to profile more than one parameter in an analysis, separate them by a semicolon, as shown in the comment in the code above. Note that profiling the likelihood can be quite time consuming.

The `TYPES.ASEswitch` turns on the calculation of asymptotic standard errors. I use a numerical approximation of the Hessian matrix, which is not very robust; it can easily return complex numbers when something goes wrong. The profiles are far more robust.

The `TYPES.POSTswitch` turns on an MCMC method (slice sampler, which is available from Matlab's Statistics toolbox) to calculate the joint posterior distribution of the estimated parameters, as well as marginal 95% posterior probability ranges for each parameter separately. No prior is used at this point (a value of 1 for each parameter). The sample from the posterior is saved into a file that has the name of your `mydata` file with `_MC.MAT` at the end. This sample can be used to produce error ellipses and confidence intervals on model predictions. These analyses can be called at the end of the `mydata` file, as explained later. Note that the MCMC procedure is quite time consuming. If you want to explore it, start with the Surv flavour, which is much, much faster than the other flavours, and thus makes a good testing ground. The value in this switch is the number of sample to draw from the posterior, so entering a `0` turns it off.

The `TYPES.Plotoverswitch` is used to plot the data from different studies in the same figure. This is not always very handy, but I left it in. I extended this switch (for now only in the GUTS flavour) to also allow plotting each exposure concentration in a separate graph. This is especially useful when the exposure situation is different pulses; plotting them into a single graph may lead to difficult-to-read graphs.

The `TYPES.PlotdoserespSwitch` turns on the calculation of concentration-response plots and iso-effect lines agaist time.

The flavour-specific switches are dealt with in the next chapters.

1.5 Initial parameter estimates, specifying which ones to fit

Every parameter has a name, default value, symbol, description, and range of allowable values. Furthermore, parameters can be estimated on normal or on log scale. The advantage of log scale is that it is easier for the optimisation routine to scan a large range of possible values (as e.g., for the elimination rate which can take a range spanning several orders of magnitude). All these things are taken care of in the function `def_pars.m`, which is called from the mydata script:

```
[par_mat,par_sel] = def_pars; % define parameters and initialise par_mat with defaults and par_sel with zeros
```

Next, initial values for the parameters can be specified:

```
% ============================================================================
%                   INITIAL PARAMETER ESTIMATES
% ============================================================================
% if there are more studies, enter a parameter value for each study:
% e.g. [1.5 2.3] for 2 studies. If there is more than one study,
% simply using [1.2] will result in the same values for all studies.

% parameters
par_mat(pospar.ke,:)   = [0.73];  % elimination rate (e.g. 1/d)
par_mat(pospar.kr,:)   = [1e6];  % damage/receptor repair rate (e.g. 1/d)
```

As the comments in the code explain, you can enter different parameter values for each study. If you enter a single value, whereas the data set contains more than one study, the same value is used as initial value for each study. If you have three studies in the data set, you can either enter 3 values or just 1 (entering 2 leads to an error). If you do not want to use a parameter in the analysis at all, you can simply remove the entire parameter line from the code. For example, if you never want to calculate a stage of damage on a particular data set, you can remove the line for `par_mat(pospar.kr,:)` from the mydata file. In that case, the default value will be used, but that should not affect the analysis.

After the initial parameter values, you have to specify which parameters are to be optimised to fit the data, and which have to be kept to the initial value:

```
% ======= selection matrix =========================
% put [0] for a fixed parameter, and [1] when the parameter must be
% estimated. When there is more than one study, common parameters can be
% indicated with -x (where x is number of the study where parameter should be copied from).
% E.g. [1 0 -1 1 -4] would mean that the parameter is estimated
% for study 1 and 4, separately; for study 2 it is fixed to the user input,
% for study 3 it is always kept at the same value as study 1, for study 5
% it is set to the value in study 4.

par_sel(pospar.ke,:)      = [1];
par_sel(pospar.kr,:)      = [1];
```

The way this selection procedure is set up allows for flexibility. You can start by fitting only a few parameters, and add parameters to the fit in a step-wise procedure (as fitting a large number of parameters simultaneously requires a very good starting position). Furthermore, if you have multiple studies, you can select common parameters between the studies. If you do not want to estimate a parameter, you can put a 1 in the `par_sel`, or comment or remove the entire line of code. Removing the line means that the default selection value will be used, which is a zero.

1.6 Last part of the script: starting the analysis

In the last part of the mydata script, the analysis starts. The actual data set `DATA` (which can be a selection of your entire data set) is run through a consistency check again. This checks for the most common errors and prints the result on screen.

```
% ======= here the analysis starts =========================

% Check the consistency of the presented dataset, in combination with the required type of assessment
cons_check(par_mat,par_sel,mfilename);

p = start_calc(par_mat,par_sel,[1 1],mfilename); % start the optimisation by calling start_calc.m
% the first [1] ensures that output graphs and estimates will be produced, [0]
% suppresses output. the filename is passed on for identifying the output.
% calling calcstart with [1 flag] is used to modify the optimisation method
% (always a Nelder-Mead Simplex search):
% 1) Two rounds: first rough, second more detailed (standard)
% 2) Two rounds: two rough ones (specifically for trying to find optimum)
% 3) One round: only a detailed one (when already close to optimum)
% 4) One round: only a rough one (for quick and dirty runs)
```

Then, the parameter values and selection matrix are turned over to the function `start_calc` which prepares for optimisation, calls the optimisation routine, handles the output to screen, and deals with any additional calculations (such as confidence intervals). At this moment, I strictly use the Nelder-Mead Simplex of Matlab (`fminsearch`), because it is part of standard Matlab, and because it is very robust in these kinds of analyses. You can change the flags to suppress the output or to change the number of optimisations. I have good experience with option 1 (a rough optimisation, followed by a detailed one), but your mileage may vary.

The very last step in the `mydata` file is the call to the functions that use the saved file from the MCMC analysis to make an error ellipse for two parameters, and posterior 95% confidence intervals on the model curves. You can do the MCMC analysis and these calculations in the same run (when `TYPES.POSTswitch` is set to a non-zero value). Otherwise, these calculations will use the saved `MAT` file from the previous run of the slice sampler (make sure it exists!).

```
% Warning: experimental!
% The next two lines use the MCMC slice sampler from the statistics toolbox
% to make an error ellipse for two parameters and plot confidence intervals
% around the model curves. Make sure TYPES.POSTswitch is set correctly.
% But, these routines are also able to work with previously stored MCMC
% samples, in the MAT file with the name of this script, plus "_MC".
% (calc_slice.m saves this file itself every time it is run)
calc_ellipse(mfilename,[pospar.c0s 1;pospar.c0s 2]); % error ellipse for two parameters
calc_conf(mfilename); % plot conf intervals on model curves
```

1.7 Types of model output

Output of the analysis is displayed in the Matlab command window. First, the results of the consistency checks will be displayed; if there are problems, an error results. If not, the optimisation starts, and the iterations of the Simplex routine are shown. The `min f(x)` value that changes all the time is the overall min-log-likelihood that is minimised (and the log likelihood is therefore optimised). After the optimisation, the results are presented. First, you get some information that is useful for keeping track of your results (filename and date), a specification of the input data that were used, the type of analysis that was performed, and whether the search routine converged. Below that, the final parameter values are displayed. Only the parameters that are judged to be 'relevant' for your analysis will be shown, but in any case, all estimated parameters are reported. A 1 or a 0 after the value shows whether the parameter was fitted or that it was not. Next to the 0 or 1, the unit is given (derived from the units you specified early in the `mydata` file). Below the parameters, results from the optional post analyses will be presented (such as confidence intervals). All of the output to screen is added to a log file `results.out` in your active mydata directory. The log file helps to keep track of things when Matlab crashes or when you want to trace back what a previous analysis produced. The log keeps on growing, so you can decide to throw this file away once in a while (a new one will be automatically created in the next model run).

Also, figures will pop up for all of the data in the data set that were used in the analysis, with the best model fit. The endpoints are plotted as function of time, for all exposure concentrations in a study in the same figure window (unless you played with the switches in the mydata file). Different concentrations are shown with a different symbol, model fits are shown as solid lines, and the line is linked to the corresponding symbols with a dotted line. In case there are replicates at a certain concentration and time point (and entered in the data set as individual values), the symbol represents the average (after the transformation

specified by the value of `lam` for each `set`), and the range is plotted as a vertical solid line through the symbol. Units in the figure window are derived from the units specified in the mydata file. In the plotting routine itself (`plot_all.m`) you can will find a few switches in the top of that file to suppress plotting of ranges, and plotting of uncorrected survival data.

The profile likelihood and posterior calculations also produce figures. For the profiles, progress can be monitored as the procedure will plot a new point in the graph as soon as it has calculated it. When the profile for a parameter is completed, a line will be drawn through the points, and a horizontal dotted line indicates the cut-off criterion (the critical value for the χ^2 distribution with one degree of freedom, at 95% confidence). The interpolated values at these cut-offs represent the confidence interval, which is given in the command window. The interval is not necessarily one continuous set; it can be a broken set, which is indicated in the output. Broken sets and ragged profiles can indicate numerical problems in the procedure. It is possible that the profile calculates ratios of less than zero. This implies that a better fit is found. After the profiling for this parameter is finished, the parameter values at the new best point will be presented in the command window (and in the log file). Every time a better value is found, the parameter set is saved to a special log file with the name `profiles_newopt.out`. So if you see that the profile is finding much better values, you can kill the profiling by pressing `ctrl-c`, and check this file to find a better starting point.

The posterior calculations only makes figures after it has finished, and does not show its progress. I have not yet found a good way to do this. Simply be patient (or stop the analysis with `ctrl-c`). The plots that are made are a multi-plot of the Markov chains for all parameters, and individual plot for the estimated marginal posterior distributions. A 95% probability region is derived in two ways, the first is by lowering a cut-off probability density (the dotted horizontal line in the graphs) until 95% of the probability is caught. This procedure does not necessarily results in equal probability in the tails. The second confidence interval is the 2.5-97.5% quantile range, which has 2.5% posterior probability in each tail. The first confidence interval can be a broken set, the second is always one set. Furthermore, this calculation produces a correlation matrix for the samples from the posterior distribution. The sample itself is returned to the file `start_calc.m` (at the end of that function), for optional further use. I have included an example to produce an error ellipse for two parameters, which can be uncommented to run it. The sample might also be used to produce confidence intervals on model predictions, which I have not implemented so far.

1.8 General analysis strategy

It makes little sense to enter a complex data set, select all switches and options and hit the run button. I suggest a more step-wise approach. After entering the data, provide some rough guesses for the parameter values, fit none of them (set all values in `par_sel` to zero, or comment them out), and make sure that none of the post calculations are switched on. Now run the script. Does it produce errors? If so, fix them first. Does it produce plots? That's great! Are the data plotted correctly, as you expect to see them? If not, check if you entered the data correctly in the mydata file. Does the model curve bear any resemblance to the data? Probably not. Try to take a second guess at the parameter values to see if you can get the model lines a bit closer to the data (this also gives you a better feel for what the parameters are doing in the model). If model and data are at least visible in the same plot, we can start to estimate some parameters. Don't estimate them all at the same time. Especially for life-cycle data, it makes sense to start with the control performance first. Do not try to fit more than 2 parameters at a time, until you get the model pretty close to the

data. Copy the fitted values from the output into the mydata file, fix these parameters, and fit 2 other ones. If you cannot get a good fit, try it again with other initial values. Only when you have the model lines pretty close to the data can you try to estimate all model parameters simultaneously.

Are you happy with the fit? That's great! Than it is time to attempt a profile likelihood for one parameter (preferably an important one, for which you are not sure about its value). This profile will show you whether there are better parameter values close by. Profiling can also be a good trick if you are not so happy with the fit, and are unsure what the problem is. The profile searches through parameter space in a certain way, which may lead you to the global optimum. If the profile looks good, it might be time to make a profile for all parameters. This might take some time (some 5-20 times more than the optimisation, per parameter that you profile), so it is a good strategy to start such a run before you leave the office. When you return in the morning, hopefully you will see some results. DEBtoxM also report the time taken since the analysis was started, so you can get a feel about how much time each type of analysis takes.

Chapter 2

General word on making modifications

2.1 Need more parameters?

If you need more model parameters than the one I have used, you can use `pospar.x1` to `pospar.x4`. These parameters have been predefined in `def_pars.m`. You can add them in the mydata file (simply copy a line for `par_mat` and `par_sel` and modify the parameter identifier to `x1` or `x2`, etc.). And, you can retrieve them in any of the functions in the same manner as the other parameters, early in each function. As soon as one of these extra parameters has a value different from zero, it will be included in the output. You can modify the text with these extra parameters and their ranges in the function `def_pars.m` (do not modify the identifier in the `pospar` part though, unless you know what you are doing).

2.2 Modifying performance of the confidence intervals

The profile likelihood determines the steps it makes in an adaptive manner. The parameters that govern the rules for these steps are in the function `calc_proflik.m`. I already have two sets of rules build in, that you can select by setting `prof_detail` to 1 (detailed and slow) or 2 (course and rapid). By default, it is set to 2.

The number of samples used for the posterior calculation is set in the `mydata` file, by the value in the `TYPES.POSTswitch`. The other parameters (such as burn-in samples) can be set in `calc_slice.m` (consult the Matlab manual for the correct syntax).

2.3 Modifying the time and concentration grids

In the file `start_calc.m`, the decision is made for the fine time grid. Standard, I use 100 time points between zero and the maximum value in the data set. Only in the GUTS flavour, I use 300 points, because I need a finer grid when working with pulsed exposures. This grid is used for the model curves that are plotted in the graphs. In the same file, I also make a fine grid for the concentrations. This is used for making dose-response plots and iso-effect lines. Standard, I use 100 concentrations between zero and the maximum in the data set.

Chapter 3

Specific features for the Surv flavour

This analysis is a simplification of the GUTS flavour (see [2]). It only works when the exposure concentration is constant, but it is much, much faster than the GUTS flavour. The reason is because this flavour applies the analytical solution (which only holds for constant concentrations). Also, no damage or receptor kinetics allowed. However, I did implement a choice between stochastic death (which is than the same as [1]) or individual tolerance as death mechanisms. I also added the option to use saturating uptake of the chemical (Michaelis-Menten type).

Because this flavour is so incredibly fast, it is an excellent way to get used to my programming style, and to explore the various options for standard errors, profile likelihoods, and Markov-Chain Monte-Carlo methods. For all other flavours this takes much more time, so you would like to be pretty sure that you are doing it correctly!

3.1 Switches in the mydata file

The Surv flavour sports an additional switch for the mechanism of death:

```
% Select mechanism for death
% ============================================================================
%         1 : pure stochastic death
%         2 : pure individual threshold
% Default for IT is log-normal distribution for now, but log-logistic can
% be set in call_deri.m
% ============================================================================

% Switches for analysis types
TYPES.dosemetric         = 3; % select the dose metric to be used (see table above)
TYPES.mechanism          = 1; % select the death mechanism to be used (see table above)
```

The `TYPES.mechanism` allows to select either pure stochastic death or pure individual tolerance.

3.2 Control mortality

The Surv flavour allows a constant hazard rate for background mortality, but also a Weibull factor. The control mortality is implemented in `call_deri.m` as follows:

$$S_0 = \exp\left(-(h_b t)^{F_W}\right) \tag{3.1}$$

When the Weibull factor F_W is 1, there is only a constant hazard rate h_b.

3.3 Limitations at this moment

Tolerance distribution

I have only implemented the log-normal and log-logistic distribution. These distributions are symmetrical after log transformation, which is why I can use a 'factor of spread' as width of the distribution. This factor is what you have to multiply and divide the median with to cover 95% of the distribution. This factor is thus easier to interpret than a standard deviation on log scale.

3.4 Making modifications

In this section, I will discuss some of the most logical places to start when wanting to mess with DEBtoxM.

Modifying the implemented models

The model calculation are performed in the function `call_deri.m`. Here, you can make modifications if needed, although this flavour is quite limited in the number of modifications allowed because of the need for using the analytical solution (the GUTS flavour allows for more tweaking).

Chapter 4

Specific features for the GUTS flavour

This analysis is based on the GUTS paper by [2], in which a range of mechanistic models for survival analysis are unified. In the code, I deviated somewhat from the symbols used in the GUTS paper. The reason is that I want to be consistent with DEB, and thus with the other DEBtoxM flavours. However, in the description of each parameter you will see the symbol as used in GUTS, in parentheses. I added the option to use saturating uptake of the chemical (Michaelis-Menten type), and I also included the option to use the receptor kinetics as presented in [6].

For the calculations of internal concentrations and damage, I strictly use the ODE solver. In some cases, this is not the most efficient approach, as analytical solutions can be applied (than use the Surv flavour of the previous chapter). However, for the sake of generality, I stick to the ode solvers here.

4.1 External concentrations as data set

The GUTS flavour (for now) is the only one that allows entering time-varying external concentrations as a data set (`type` 0). This allows using an external model to provide exposure patterns, or you can enter measured values. If there is a set with this type in the data set, all of the data sets in this study will use these time-varying exposures, instead of the values in the vector `C` (which serves as a kind of 'name' for each exposure scenario). In general, it is safest to only analyse effects data for which you specified an external concentration data set. If the effects data apply a value in the concentration vector `C` that does not occur in the external concentration data, the function `start_calc.m` will estimate the missing concentrations using the available data and the vector `C`. This only works properly when the values in `C` are representative values, e.g., nominal initial concentrations. The estimated external concentrations are plotted with the ones in the data set as broken lines. There should be only one data set with external concentrations per study, to avoid confusion (DEBtoxM will return an error in the consistency check if you attempt it anyway). See also the code example on Page 5.

The external concentration set is not fitted with a model; the concentrations used in the model will be interpolated at small time steps. For this interpolation, I use a piecewise cubic Hermite spline (`pchip` in Matlab). I generally prefer this one over a linear interpolation as it attempts to fit a continuous curve through the points (which is easier on the numerical procedures), and it does not add a lot of extra dynamics in doing so. These interpolations

on the external concentration will be presented in graphs too, so you can judge whether the interpolation works as expected. Furthermore, if you use an external concentration set without measured internal concentrations, I also produce a multi-plot with the estimated (scaled) body residues in time (solid line), scaled damage levels (when relevant; dashed line), and the NEC for survival (dotted line). This could help in the interpretation of the survival patterns (especially when something goes wrong).

If you do not enter an external concentration data set, the analysis will use the value presented in the vector `C` for each data set, as a constant value in time.

4.2 Switches in the mydata file

The GUTS flavour sports more options for the dose metric, compared to the other flavours. Furthermore, it includes an extended choice between mechanisms of death compared to the Surv flavour (also allows a mixed approach):

```
% ========================================================================
% Select dose metric to be used (M)
% ========================================================================
%         1 : external concentration (no toxicokinetics)
%         2 : scaled internal concentration
%         3 : actual internal concentration
%         4 : scaled damage level
%         5 : fraction occupied receptors
%
% In principle, dose metric 4 and 5 are based on the actual internal
% concentration, and thus require measured body residues or a good
% estimated for PVd and ke. The scaled internal concentration can be used
% by fixing PVd=1 (but this might lead to problems identifying ke and kr).
%
% Select mechanism for death
% ========================================================================
%         1 : pure stochastic death
%         2 : pure individual threshold
%         3 : mixture (stochastic death with distributed threshold)
% Default for IT is log-normal distribution for now, but log-logistic can
% be set in call_deri.m
% ========================================================================

% Switches for analysis types
TYPES.dosemetric         = 3; % select the dose metric to be used (see table above)
TYPES.mechanism          = 1; % select the death mechanism to be used (see table above)
```

The `TYPES.dosemetric` allows you to select the dose metric (M in GUTS terminology). This allows to use the four dose metrics of the original GUTS, and the (slightly different) receptor kinetics approach of [6]. Based on which dose metric you select, some parameters will have no effect in the analysis (and will not be reported). For example, if you do not use damage of receptor kinetics, the 'repair rate' k_r will have no effect on the analysis. The damage and receptor kinetics methods assume that you have included a data set with internal concentrations, or good estimates of toxicokinetics (thus, for the elimination rate k_e and the bioconcentration factor P_{Vd}). If you want, you can still mimic the use of scaled body residues together with damage or receptors by fixing the bioconcentration factor to 1. The only glitch is that now the units in the output are not correct anymore (the unit for 'internal concentration' should be replaced by that for 'external concentration').

When the dose-metric is not 1 or 2, the NEC (z in GUTS terminology) does not have the interpretation of an external concentration without effects. Also, when you use saturating

intake of the chemical into the body, the NEC looses this interpretation. In these situations, I make a recalculation of the (median) external effects threshold and report it below the parameter estimates in the output.

The `TYPES.mechanism` allows to select either pure stochastic death or pure individual tolerance, or the GUTS mixture. In principle, the latter would suffice, as you can use parameter values to go continuously from one to the other. However, researchers will often be interested in these special cases. In DEBtoxM, somewhat different methods are used for each of these three options to optimise calculation efficiency.

4.3 Control mortality

The GUTS flavour allows a constant hazard rate for background mortality, but also a Weibull factor. The control mortality is implemented in `call_deri.m` as follows:

$$S_0 = \exp\left(-(h_b t)^{F_W}\right) \tag{4.1}$$

When the Weibull factor F_W is 1, there is only a constant hazard rate h_b.

4.4 Limitations at this moment

Tolerance distribution

I have only implemented the log-normal and log-logistic distribution. These distributions are symmetrical after log transformation, which is why I can use a 'factor of spread' as width of the distribution. This factor is what you have to multiply and divide the median with to cover 95% of the distribution. This factor is thus easier to interpret than a standard deviation on log scale.

4.5 Making modifications

In this section, I will discuss some of the most logical places to start when wanting to mess with DEBtoxM.

Modifying accuracy of the calculations

Some calculations use an ODE solver. I use this for the calculation of internal concentration, damage and receptors, and hazard. However, hazard is only calculated with the ODE solver when using pure stochastic death! When the threshold is distributed, I would have to call the ODE solver for every sample from the distribution, which is not very efficient. When using this mechanism of death, I calculate body residues and damage/receptors with the ODE solver on a small time grid, and calculate the hazard rate (or survival probability) from that.

The precision of the ODE solver can be set in the function `call_deri.m`; I have set the tolerances tighter here than the Matlab defaults, which improves the results (in my opinion), but slows down the calculations. Standard, I use the solver `ode15s` here. This solver should be better suited for stiff cases (e.g., when you have pulsed exposure). However, I found that it takes such enormous steps that it might completely miss a second pulse! Thus, I decreased the upper limit for the step size to the maximum time divided by 100. If you think that is not enough for your situation, adapt the `options` call in `call_deri`. In this Matlab function,

I also included the option to use an events function to catch cases where the derivatives are discontinuous (e.g., when a NEC is crossed). At this moment, I am not so sure whether this is useful or needed, so I commented it out.

In the file `start_calc.m`, the decision is made for the fine time grid. Standard, I use 300 time points between zero and the maximum value in the data set. In the GUTS flavour, this grid is used for splining of the external concentrations, the individual tolerance calculations, but also for the model curves that are plotted in the graphs.

Modifying the implemented models

The models for internal concentrations and damage/receptors are located in the function `derivatives.m`. Here, you can include your favourite models. If you need more model parameters, you can use `pospar.x1` to `pospar.x4`. These parameters have been predefined in `def_pars.m`. You can add them in the mydata file (simply copy a line for `par_mat` and `par_sel` and modify the parameter identifier to `x1` or `x2`, etc.). And, you can retrieve them in the `derivatives.m` in the same manner as the other parameters, early in this function. As soon as one of these extra parameters has a value different from zero, it will be included in the output. You can modify the text with these extra parameters and their ranges in the function `def_pars.m` (do not modify the identifier in the `pospar` part though, unless you know what you are doing).

You can also include your own model for the external concentration in the file `derivatives.m`, as long as it is time explicit (a simple function of time). You can use the predefined extra parameters `pospar.x1` to `pospar.x4`. For example, if you have exponential decay of the external concentration, add early in `derivatives.m`:

```
kdeg = p2(pospar.x1);  % degradation rate in medium
c = c * exp(-kdeg*t);  % recalculate c for an exponential decay
```

If you want to modify the calculations for survival probability, take care that for pure stochastic death, the hazard rate is calculated using `derivatives.m`. When the mechanism includes a tolerance distribution, the calculation is performed in `call_deri.m`. Again, you can modify the models, and add extra parameters, in the same way as explained above.

Modifying the spline function

Standard, I use the piecewise cubic Hermite interpolation (`pchip`). If you prefer a different type (e.g., `linear`), you can adapt these interpolations. To this end you need to change the interpolation in the file `start_calc.m` (around Line 130). Note that there are two versions of the interpolation method included here as Matlab changed several functions and I want to be compatible with the older versions.

Chapter 5

Specific features for DEBtox flavour

The DEBtox flavour is a simplification of the DEB3 model as published in [10]. The simplification is comparable to the old DEBtox model from [9], but it repairs a few errors, works with actual instead of scaled length, and includes the dynamics of the reserves (which does not require additional parameters). Compared to DEB3, the simplification is caused by removing maturity as a state variable (it is assumed to be proportional to structure, in all cases), and by setting the egg costs constant. The resulting parameters are mostly easy-to-interpret compound parameters such as the maximum reproduction rate and the Von Bertalanffy growth rate constant.

5.1 Switches in the mydata file

The DEBtox flavour sports several switches that do not occur in (some of) the other flavours. The first is to select the type of blank mortality (although there is not yet much to choose from). Because this flavour also deals with sub-lethal effects, you have to select a mechanism of action (the affected DEB parameter):

```
% ============================================================================
% Types of blank mortality
% ============================================================================
%         0 : no background mortality
%         1 : constant hazard rate (DEBtox standard)
...
% Types of affected DEB parameters (TmoA)
% ============================================================================
%         0 : no effect at al
%         1 : increase in maintenance (somatic and maturity)
%         2 : decrease in ingestion or assimilation
%         3 : increase in costs for structure and maturation
%         4 : hazard to the embryo
%         5 : costs for reproduction
%         6 : costs for structure, maturation and reproduction
% ============================================================================

% Switches for analysis types
...
TYPES.blanktype          = 1; % type of blank mortality/ageing (0 is 'off')
...
[par_mat,par_sel] = def_pars; % define parameters and initialise par_mat with defaults and par_sel with zeros

for study = 1:TYPES.nstudy,
    ADD(study).TmoA = 2;   % Toxic Mode of Action for each study; if needed, each study can be given a different TmoA
    ADD(study).Tovi = 0;   % for oviparous species, time between egg production and counting of the offspring
    ADD(study).Th   = 0;   % hatching time of eggs for population calculations (NOT for oviparous species!)
end
```

You can select a different mode of action for each of the studies you want to analyse, but the way it is shown above, the same one will be used as default. Additionally, you can specify two time durations, `Tovi` or `Th` (the first will override the second). `Tovi` is used when the species you investigate is oviparous, i.e., hatches its eggs internally, such as in *Daphnia*. The observations on reproduction will represent the emergence of hatchlings whereas DEB treats the production of the egg as reproduction. Internally, the observed reproduction data will be shifted `Tovi` time units (same unit as specified for the entire data set) for the comparison to the model predictions. The plots will show the reproduction data on the normal time scale. `Th` can be used to specify hatching time for eggs, which is only used for the calculation of population growth.

The `TYPES.Popswitch` turns on the calculation of the intrinsic rate of increase (using the Euler-Lotka equation, see [3]:

```
% Switches for standard errors and confidence intervals
TYPES.PopSwitch          = 0; % switch for calculating population growth rate (1 is 'on')
```

This will calculate the population growth rate as a function of the exposure concentration, at several food levels. You can set these food levels in the file `calc_popmod.m`. When using the Euler-Lotka equation, the question is over which time interval to integrate survival and reproduction. For example, a *Daphnia* reproduction test lasts 21 days. Integrating over 21 days assumes that all individuals die immediately at $t = 21$ (or at least stop reproducing completely). Integrating up to infinity would assume that reproduction and survival continue just like they did over the first 21 days, which is also unrealistic. Fortunately, the first clutches of offspring have the highest impact on the population growth rate (because of interest-upon-interest). As a pragmatic solution I implemented a constant factor of two on the test duration; so for *Daphnia*, I integrate over 42 days. This factor can be modified in the top of the function `calc_popmod.m`. More realistic analyses require inclusion of an ageing module.

5.2 Control mortality

The DEBtox flavour allows a constant hazard rate for background mortality, but also a Weibull factor. The control mortality is implemented in `call_deri.m` as follows:

$$S_0 = \exp\left(-(h_b t)^{F_W}\right) \tag{5.1}$$

When the Weibull factor F_W is 1, there is only a constant hazard rate h_b.

5.3 Limitations at this moment

Modes of action

I have only implemented 6 different modes of action. These were selected carefully to not interfere with the constant length at puberty (remember that maturity is not followed explicitly in this flavour). If you see that the length at puberty varies with toxicant exposure, you might want to use the DEB3 flavour to analyse the data properly. Several mechanisms of action change the length at puberty and therefore cannot be implemented (e.g., a change in κ).

Ageing

I published papers in which I combine a DEBtox simplification with an ageing module. However, I am not too confident in their validity. For the time being, I have implemented a simple alternative ageing model, which is described in detail in the technical document that can be found on http://www.debtox.info/book.php.

Contribution of reserves

The contribution of reserves to total body weight or volume is ignored, just like the effect of reserves on toxicokinetics.

5.4 Making modifications

In this section, I will discuss some of the most logical places to start when wanting to mess with DEBtoxM.

Modifying the implemented models

The model calculation are performed in the function `derivatives.m`. Here, you can make modifications if needed, although you have to watch out to maintain consistency.

Bibliography

[1] J. J. M. Bedaux and S. A. L. M. Kooijman. Statistical analysis of bioassays based on hazard modelling. *Environmental and Ecological Statistics*, 1:303–314, 1994.

[2] T. Jager, C. Albert, T. G. Preuss, and R. Ashauer. General Unified Threshold model of Survival - a toxicokinetic-toxicodynamic framework for ecotoxicology. *Environmental Science & Technology*, 45:2529–2540, 2011.

[3] T. Jager, T. Crommentuijn, C. A. M. Van Gestel, and S. A. L. M. Kooijman. Simultaneous modeling of multiple endpoints in life-cycle toxicity tests. *Environmental Science & Technology*, 38:2894–2900, 2004.

[4] T. Jager, E. H. W. Heugens, and S. A. L. M. Kooijman. Making sense of ecotoxicological test results: towards application of process-based models. *Ecotoxicology*, 15:305–314, 2006.

[5] T. Jager and C. Klok. Extrapolating toxic effects on individuals to the population level: the role of dynamic energy budgets. *Philosophical Transactions of the Royal Society B-Biological Sciences*, 365:3531–3540, 2010.

[6] T. Jager and S. A. L. M. Kooijman. Modeling receptor kinetics in the analysis of survival data for organophosphorus pesticides. *Environmental Science & Technology*, 39:8307–8314, 2005.

[7] T. Jager, T. Vandenbrouck, J. Baas, W. M. De Coen, and S. A. L. M. Kooijman. A biology-based approach for mixture toxicity of multiple endpoints over the life cycle. *Ecotoxicology*, 19:351–361, 2010.

[8] S. A. L. M. Kooijman. *Dynamic Energy Budget theory for metabolic organisation*. Cambridge University Press, Cambridge, UK, third edition edition, 2010.

[9] S. A. L. M. Kooijman and J. J. M. Bedaux. Analysis of toxicity tests on *Daphnia* survival and reproduction. *Water Research*, 30(7):1711–1723, 1996.

[10] S. A. L. M. Kooijman, T. Sousa, L. Pecquerie, J. Van der Meer, and T. Jager. From food-dependent statistics to metabolic parameters, a practical guide to the use of dynamic energy budget theory. *Biological Reviews*, 83:533–552, 2008.