Short manual for the openGUTS Matlab prototype

Tjalling Jager*

May 17, 2019

This documents is part of the openGUTS project, and can be downloaded from http://openguts.info/. The openGUTS project is made possible by funding from Cefic-LRI in project ECO39.2.

Contents

*DEBtox Research, De Bilt, The Netherlands. Email: tjalling@debtox.nl, http://www.debtox.nl/.

1 Introduction and workflow

This documents provides a short user manual for the Matlab prototype of the openGUTS software. The prototype was developed by DEBtox Research, taking elements of the well-tested BYOM package (http://debtox.info/byom.html) and placing them into a GUTS-only framework. The Matlab prototype serves as the platform for developing and testing functionality and algorithms for the GUTS software, and thereby acts as the blueprint for the C++ code, produced in this project by WSC Scientific GmbH. This prototype should aid testing and future development of the GUTS software, and can be used to test model extensions. Note that this prototype is distributed under under the terms of the GNU General Public License, version 3. This document assumes knowledge about the GUTS concepts, following the e-book [2].

Relevant general points:

- The workflow of openGUTS is based on the proposed workflow in the EFSA scientific opinion on TKTD models for use in risk assessment of pesticides [1]. However, (elements of this) workflow will be useful for many other purposes as well (including scientific research). For example, openGUTS can also be used to calculate classic LC$x, t$ values.

- The Matlab prototype and the standalone software will have the same basic functionality. However, the prototype has much less of a user interface, does not produce formatted output reports, and does not allow saving/modifying data sets (this should be done in Excel or in a text editor). The protoype has some extra functionality that is not built into the standalone version (e.g., the option to plot confidence intervals on survival curves in the LPx predictions).

- Running the prototype requires Matlab. None of the toolboxes are used. The prototype (v0.3 and later) has been made compatible with older versions of Matlab. It has been tested with versions R2015a, but should also work with R2014b (*not* with R2014a!). Most recent version tested is R2018a.

- The prototype and the software will only include the simplest cases of the model: GUTS-RED-SD and GUTS-RED-IT.

- The special cases for fast and slow kinetics are not included (see the GUTS e-book [2], Appendix C). Instead, a default range for k_d is used, which will basically be the same for all data sets. The user will get a warning when k_d (or any other parameter) runs into a boundary.

- The same input data format is used for the calibration and validation stages (for exposure profiles, it will be different). The prototype uses the same text file format as the standalone software, so input files can be used by both versions. However, the project files will *not* be compatible.

- The Matlab GUI for opening files is used, which makes it easy to work with saved files.

The general workflow of the software is presented in Figure 1. Text files will be used as input (the standalone version will also allow working with an input grid). The results of a calibration are stored in a 'project file', which the remainder of the workflow will load when needed. Note the 'load calibration', which implies skipping the first sections (loading input data and the calibration itself), loading a previously saved project file, and plotting the results from that.

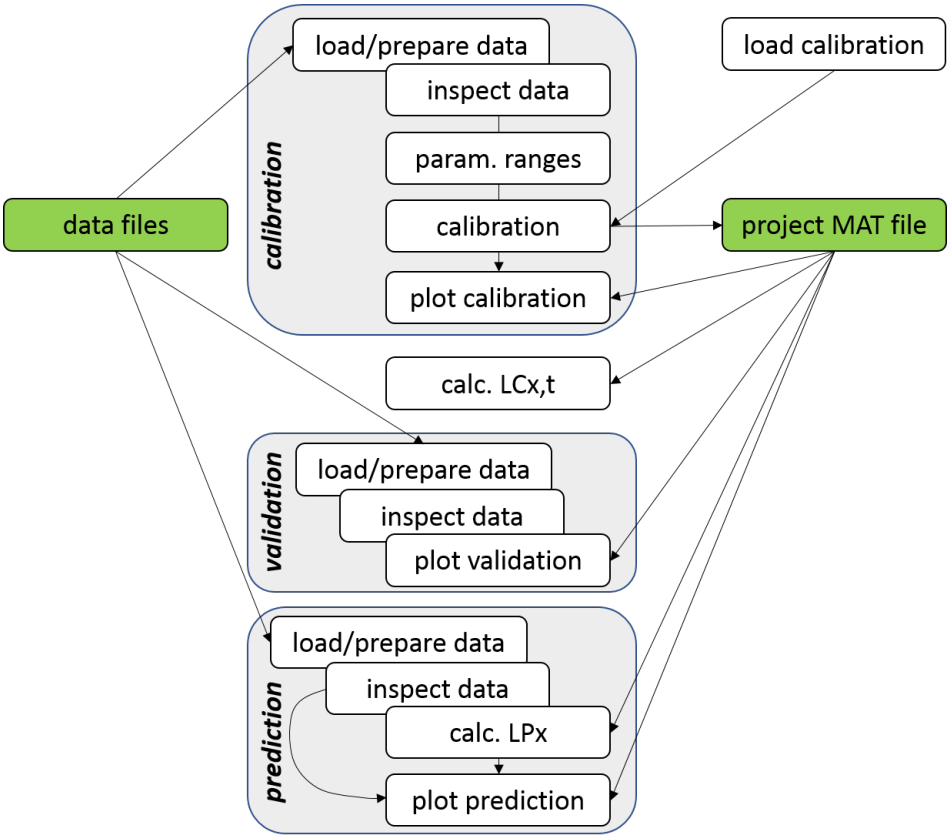

Figure 1: General workflow of the software. Blue fields indicate main parts of the workflow of the software: calibration, validation, prediction. The LC$x, t$ calculations a shown separately as it does not use data input. The direct arrow in the prediction stage, from 'inspect data' to 'plot prediction' could be used to plot model predictions for user-provided multiplication factors (this will not be part of the standalone software).

2 Installation and quick-start guide

2.1 Directories and file types

Installation. Unpack the `zip` file in a location of your choice on your computer. Make sure that the top-level directory is `openGUTS`, and don't change this name. It is *very important* to maintain the name of this folder as it is used to place the directory `engine` into Matlab's search path. The `engine` directory contains all of the functions needed for the openGUTS calculations (in general, there will be no need to modify these files).

At this moment, the prototype is set up with a single analysis folder, named `protoguts`, which contains the basic script `protoguts.m` (this is the only Matlab file you need to modify/run). This basic script follows the general workflow of the software. This folder must be under the `openGUTS` folder, and will contain a number of sub-folders (if they are not there they will be automatically generated when running the script):

- `input_data` contains the input data files for survivors/exposure scenarios. Each file contains one data set (with a number of treatments) in a formatted manner. These are plain text files (tab delimited) that can be opened/modified/saved by Excel or NotePad. You can use the sample files distributed with this prototype as template.

- `input_profile` contains the input data files for exposure profiles used in predictions (LPx). Each file contains one data set (with a single treatment). These will thus be plain text files with two columns (tab delimited): time points and concentrations (and no headers or other information).

- `output_sample` will contain the binary `mat` files with the information from the parameter-space explorer such as the best-fitting parameter set and the sample from parameter space. Plotting and post-analyses (such as calculation of LC$x, t$ and LPx) will use these saved files. The saved files will have the name of the analysis (as defined in the script from which they were started), with text added to clarify the special case (`_SD` or `_IT`). This file has some of the functionality of the 'project file' of the standalone version, but does not include any model output other than results from the calibration.

- `output_report` will contain all of the output plots (as PDFs) and a log file that collects all screen output (cumulatively!). All saved files will have the name of the analysis, and a specification of what they are.

Note that the output directories can become pretty full when working with the prototype for a while, which takes up a lot of disk space. Regularly remove things that you don't need anymore. The input data format will be the same for the Matlab prototype and the final standalone software, so an input file should run on both platforms in the same manner.

You can add sub-directories under the `openGUTS` folder with your scripts for various projects if you like. Simply copy the files `protoguts.m` and `pathdefine.m` to your new folder, and create a new folder `input_data` and `input_profile`.

2.2 Quick start

General remark: working in steps. You will generally only work with the script `protoguts`. This script is set up to run in sections, rather than through the entire workflow automatically. In the script, you will see several `return` statements, which will stop the analysis at that point. Thus, when first run, the script will stop after calibration (`BLOCK 2`) and before calculation of the LC$x, t$ (`BLOCK 3`). This gives you the opportunity to check whether the calibration did its job before continuing with further analyses. After a successful calibration, you can run the subsequent parts (`BLOCK 3-6`) in any particular order: simply position your cursor in the Matlab editor within a block of code you like to run and press 'Run Section', or select the code from the block you like to run and press `F9`. Also, you can remove one or more of the `return` statements, and/or comment out blocks of code. All of these blocks use the saved information from the binary `mat` files, but don't write to it.

General remark: keeping track of what you do. In `BLOCK 1`, you can set a name for the analysis that you are doing. This name will be used as header in the plots, in the output to screen, and for the different files that are saved during the analyses. After running the calibration, take a look in the directories `output_sample` and `output_report` at the files that are produced. Avoid spaces in this name, or Matlab may complain at some point.

Getting started with calibration. Run the script `protoguts.m`, select the input file `propiconazole_constant` and see what happens. A series of plots is produced: an inspection plot of the input data file, a parameter-space plot that shows the progress of the optimisation, and final fits. The inspection plots are meant to provide visual feedback to whether the right data are entered in the right manner (you can place a `return` statement after the first plot call in `BLOCK 2.2` to force Matlab to stop, otherwise calibration will start immediately).

Additionally, information is printed on screen in the Matlab command window. After plotting the fit, the analysis stops, due to the `return` statement that I added in the script file `protoguts` at the end of `BLOCK 2`. Calibration is pretty rapid: expect 30 seconds to 5 minutes, depending on the data set and your PC. Constant exposure is faster than time-varying, and calculation time depends on the number of treatments (so *combine* replicates, rather than inputting them as separate treatments!).

Background hazard rate. In the calibration, the background hazard can be fixed to a value fitted on the control treatments (identified by the header `Control` in the input file). To do this, set the global variable `GLO.fix_hb_cal` to 1 (in `BLOCK 1` of the code). Also for validation, background hazard can be fixed to the value in the validation data by setting `GLO.fix_hb_val` to 1 (`BLOCK 4`).

Validation with other data. For validation, run the code in `BLOCK 4` after successful calibration. For example, use the file `propiconazole_pulsed_linear` for validation. Note

that the concentration unit must be the same as in the calibration data, otherwise an error is produced.

Running SD and IT. The code is set up in such a way that calibration and validation are done for both SD and IT (sequentially), and LC$x, t$ calculation as well. For the prediction of LPx (using exposure scenarios in separate input files) a choice needs to be made between SD and IT by setting the variable `sdit` (`BLOCK 5.2` and `BLOCK 6.2`). However, the code can be easily modified to your taste.

Predictions for new exposure scenarios. Prediction of LPx (`BLOCK 5`) requires an input file with an exposure scenario; several files are included in the prototype (a.o., the FOCUS profiles used for the GUTS case study in the EFSA opinion[1]). Calculation of LPx with its confidence interval may be very time consuming for FOCUS profiles (15 minutes for IT and about 1 hour for SD, depending on your PC). Batch processing (`BLOCK 6`) is done without calculating CIs so this is pretty fast (about a second per profile). Note that the comments explain how individual LPx can also be calculated without confidence interval.

Input data files. For the format of the input data files, take a look at the samples in the directories `input_data` and `input_profile`. These are simple tab-delimited text files which can easily be read and modified with Excel. The Matlab version saves an internal data representation to a binary project file (extension `mat`), together with the calibration results. All fits and post-calculation will use this file, but this file is not compatible with the project file of the standalone version.

The example files with input data should be self-explanatory. Note that the files `propiconazole_pulsed_linear` and `propiconazole_pulsed_renewals` represent the same data set, but define the exposure scenario in a slightly different manner. The second version uses an instant change in concentration by entering two concentrations at the same time point (this is allowed and slightly more efficient than adding time points as done in the first data set). Note that missing data can be entered as a '-' (minus sign), though the Matlab prototype will interpret any non-numerical entry as a missing value.

Changing default parameter settings. The parameter ranges and the flags for fitting/fixing a parameter and for log/normal scale fitting are in the variables `pmat_SD` and `pmat_SD`. First column is not used, second column is fit (1) or fix (0), third and fourth column are the range that is searched, and the fifth column is normal scale (1) or log-scale (0). These settings are all prepared automatically by the call to `startgrid`, so there is usually no need to meddle with them. However, if you need to change something, you

[1] In the EFSA opinion, there is some confusion regarding the concentration units. The calibration data are reported as nmol/ml (or μM) whereas the exposure profiles are in μg/L. In the files distributed with the prototype, this was not changed, to allow the users to reproduce the same numbers as in the opinion. However, in the script file (protoguts, BLOCK 6.1), there is the possibility to set a correction factor. Nevertheless, these files are examples only, and not meant to be used for research or regulatory application.

can break the analysis after `BLOCK 2.2` and change the parameter matrices manually. You can also add some code at this point to modify some values. Note that there is a global with the position in the matrix for each parameter. You can thus refer to them by name. For example, m_w is in the second position, so `GLO.mw=2` and we can fix it to a value of 10 for the SD analysis by setting `pmat_SD(GLO.mw,[1 2]) = [10 0];`. Note that all lower bounds should be above zero.

Warning: changing parameters ranges away from their default values is done at your own risk! some settings will lead to poor performance or even complete failure of the calibration.

Returning to a previous calibration. Once a calibration is performed, there is no need to repeat the calibration to make new predictions. It is always possible to work from (or show) a previous calibration by setting `use_saved_sample` to 1 (`BLOCK 1`) and running the script. That will read the saved information for a previous analysis, and also use the name saved with the previous analysis for all output (so it will overwrite output you made previously with this name!). Running the script with `use_saved_sample=1`, you will be prompted to select a `mat` file from a previous analysis. Select either the SD or IT file, and both will be loaded and shown.

3 Final words

3.1 Some workarounds

For specific types of analysis, it is possible to apply a workaround. However, it is good to note that the BYOM platform includes more types of models and more flexibility in the types of analysis that can be done (at a cost of decreased user-friendliness).

Mimic the full model. To some extent, users can mimic the full models GUTS-SD and GUTS-IT by calculating the body residues themselves (e.g., with a one-compartment model) and entering them as a time-variable exposure scenario for the reduced models. This should work well enough, although he uncertainty in the body residues cannot be propagated through to the survival probability anymore.

Missing individuals. It is good to note that we can also accommodate experiments where individual animals went missing or were removed during the test (e.g., for body-residue analysis). The simple solution is to split up the treatment in two (which need a different identifier). The animals that are removed/missing at a certain point get their own treatment: they are counted as survivors as long as they are in the test, and as 'missing data point' after they have removed/went missing. This is a bit of work when creating the input data set, though.

Validating on LC50s. Users can also validate on LC50s by entering them as a data set with constant exposure and some fake data: e.g., 10 survivors at $t = 0$ and 5 survivors at the time point at which the LC50 was determined. Every LC50 would then need to be a new treatment. The only thing to note is that the CI on the data point will be meaningless.

3.2 The Matlab publishing option

Matlab contains the possibility to make nicely formatted output files for an analysis. In the editor, check out the publishing options. For example, the option to make a Word document works well. You can decide to include the code or not, and each section in the code becomes a new section in the Word document. You can play around with the script and the publishing options to get the kind of report you like. Also, consider the possibility to make separate scripts yourself, with parts of the analysis. For example, you can have one script that calibrates and validates the model, and a separate one that loads a previously-saved analysis and makes predictions for FOCUS profiles. I suggest to use the following settings in the publishing options:

Output file format	doc
Figure capture method	entireGUIWindow
Include code	false
Evaluate code	true

Note: the comments in the script file appear in a formatted form in the output report, at least when they are part of the block immediately following the `%%` sign signalling a

new section of code. Everything between <> is interpreted as a hyperlink. In the code for the prototype, I follow the convention to use these <> to indicate variables or functions in the comments. To avoid getting silly hyperlinks in your report, I changed the main script `protoguts.m` to use [] rather than <> for variables and functions. This is *only* done in this script, so when you `include` a function in your output report, you will get these nasty nonsense hyperlinks again.

3.3 More information

For the prototype, there will be a design document with more explanation on the underlying models and on which file does what. We will also prepare a document to aid the interpretation of the software's output, with several example data sets that demonstrate specific deviating behaviours. However, I hope that for this stage of the project the current quick start is sufficient, together with the extensive comments in the code. The documents will be posted on http://openguts.info.

For questions and comments, I would appreciate if you use the discussion forum at http://board.debtox.nl. This makes sure that others can see your questions, and the answers provided, which hopefully minimises email traffic (and helps other that run into the same issues). However, please note that this is *not* a help desk. To post on the discussion board you need to register with an email address, and I have to manually approve you. This is done to keep the spambots out. If you have problems registering just send me an email and I will create an account for you.

References

[1] EFSA. Scientific opinion on the state of the art of toxicokinetic/toxicodynamic (TKTD) effect models for regulatory risk assessment of pesticides for aquatic organisms. *EFSA journal*, 16(8):5377, 2018.

[2] T. Jager and R. Ashauer. *Modelling survival under chemical stress. A comprehensive guide to the GUTS framework*. Toxicodynamics Ltd., York, UK. Available from Leanpub, https://leanpub.com/guts_book, Version 2.0, 8 December 2018, 2018.