**D2D Development Team Meeting**

Oberstdorf, January 2-4, 2017  
Participants: Clemens Kreutz, Bernhardt Steiert, Helge Hass, Andreas Raue

**Meeting minutes:**

* **Folder structure.** We revised the folder structure of the code and examples. Every folder should now contain a brief Readme.txt that explains what the folder is used for. (done)
  + The **root folder** now contains only essential functionality that will be used for most of the applications.
  + **Development**, contains code that is still unfinished or experimental.
  + **Subfunctions**, contains functions that should not be called directly.
  + **Advanced**, contains additional functionality that is only used in special circumstances and requires some degree of expert knowledge.
  + **Ccode**, contains the c code that will be used for compilations by the MATLAB functions
  + **Latex**, contains the Latex code that will be used by arReport and other report functions.
  + **Clusterfunctions**, extensions of normal functionality for MATLAB clusters.
  + **Deprecated**, functions that are not recommended to be used any more.
  + **ProjectTemplate**, empty project files used in arCreateProject.
  + **Matlabtools** (former arTools), additional helper code that is not depended on the remaining toolbox code.
* **Objective function calls**. Use branch! Introduce function arGetMerit that will return the objective function value that is currently selected. This function will be used through the code when objective function values are requested. Replace arChi2 by arMerit that will inherit its functionality but will also make use of arGetMerit and will be improved. Clean up C code that performs part of the merit function calculation (residuals, add 50 and Bessel correction) and add this as MATLAB code to arMerit. (**Clemens**, Berni, Helge)
  + fit\_errors: Make default suggestion, handle error model vs. std from data file correctly (default 0 = mixed, if std in data file use this, otherwise error model; 1 = use error model only). Remove matrix implementation of fit\_errors (also ndata\_err), which is redundant then.
  + Output for console and plots, show Chi^2, -2\*LL, -2\*LL (Bessel corrected)? If error parameters are fitted, show always LL. Output label in arGetMerit for plots and console.
  + plot\_errors: As for fit\_errors (default 0 = mixed, if std in data file use errorbar, otherwise error model bands; 1 = use errorbar only; -1 = predictions bands)
* **Error model.** Output warning if relative error in log10 space is used and suggest alternative formula. (Helge)
* **Revise log10 of sensitivities.** Clean up c code vs arSimu. Where does the log transformation happen? For sxExpSimu and syExpSimu everything should happen in arSimu. For sxFineSimu backtransformation in arSimu will be removed (check for usage of sxFineSimu in code and ensure compatibility). After arSimu finished, all sensitivities should be log or normal as the parameters are. (Helge)
* **Revise PPL.** Wait for arMerit and arGetMerit. (Helge)
* **Revise PLE.** Wait for arMerit and arGetMerit. (Clemens)
* **arSimu arguments**. Check is works correctly. Coordinate with arMerit arguments. (Joep, Berni)
* **Automatic benchmarks**.Fix doTest (possibly rename and place in Advanced folder) and figure out how automatic benchmarks could be implemented. (Joep)
* **Cluster functions and parallelization.** Document functions and make recommendations. (Andreas)
* **arLoadModel and arLoadData.** Split reading data from def files, filling into ar struct and additional calculations into separate sub-functions. (Clemens)
* **Change compilation of conditions.** Use branch! We assessed feasibility and concluded that it is possible but a lot of work and potentially breaks a lot of functionality. It would be possible on an experimental branch. Test case are required to check that the new implementation works for quite some time. Because of the large effort and minor advantage in the common daily work, it is only reasonable if there is a strong limitation in a working project. There are benefits beyond faster compilation times that make this change attractive. We discussed how to implement this. One way is to remove hard coding of of p’ = f(p) in c code and do this in arSimu. But is also require the back transformation for the derivatives. A first step would to remove derived and observable calculations from c code and implement in arSimu. This would make it easier to implement but also make the code slower for larger observable or derived expression that do sometimes occur. (Andreas, all)
* **Object oriented programing / API.** We discussed cost benefit and decided to keep the global ar struct which seems to be still the best solution. (deferred)
* **Search for mini functions that are used multiple times.** We did this successfully for my\_subs and arSubs. Do we need to create additional separate mini functions in the Subfunctions folder and use them through the code? We discussed cost benefit and decided that the current situation is acceptable. (deferred)