**Documentation**

ADAPT version: R&D\_0.0

*Summary of changes:*

This version is a modified version of an iteration of ADAPT produced by Yvonne Rozendaal. This version is an attempt to incorporate pre-existing ADAPT functionality (gene expression) from another ADAPT version and to add some new functionality while staying close to the philosophy behind this iteration. That is, make ADAPT run by putting all necessary information in an 'initialize\_model'-file.

*Addition of pre-existing functionality:*

- Gene-expression data can now also be included in the model.

*Addition of new functionality:*

- Performing a (local) sensitivity analysis.  
- Using results from ADAPT, i.e. parameter trajectories, as input for another run of ADAPT.  
- Comparing two experimental conditions (datasets) making use of the same model topology by either substituting parameter trajectory predictions from one condition for the other and observing the effect on the state / flux of interest.  
- Doing the same by modulating the parameter trajectory prediction by multiplying with the average ratio of parameter trajectory predictions between the two conditions (reference and target) over a specified time span.

*Other changes:*

- Now supports parallel computing.  
- Also collects the 'average flux' (R.jm) over the time-step within an ADAPT iteration and not just the flux at the end of the time-step, i.e. when the simulation for that time-step is almost at steady state. This average flux becomes important when you want to explain changes in a state that readily reaches steady state. For example: VLDL-TG production - VLDL-TG clearance should represent the change in VLDL-TG. However, when you sample both fluxes when they have reached steady state in the simulation, both fluxes will be the same and thus falsely predict that VLDL-TG should not change. This can be remedied by using the average flux of the time-steps instead.  
- Possibility to make initial conditions (states) the same as determined by the sampled point from the raw data. This may not really be necessary, ... but at some point I thought it might.

*To do:*

- Sensitivity analysis is slow. Would be much faster if the steady states could be calculated with the analytical solution. I think it should be possible to make a script that automates this calculation by doing Gauss-Jordan elimination, manipulating the equations as strings.

- Comparing two conditions, by substituting or modulating parameter trajectories now requires a lot of manual steps (i.e. copying and modifying files). This should be made easier and more straightforward.

*How to use:*

You can still run the analysis by just modifying the 'initialize\_model' file. If you want some parameters to be determined by what you provide as input, you have to define them the same way you would define parameters, but replace the 'p' with 'u'.

Thus,

m.info.p{1} = {'k\_FC\_synt' 'hepatic de novo FC synthesis ' ''};

would become,

m.info.u{1} = {'k\_FC\_synt' 'hepatic de novo FC synthesis ' ''};

If you do use input, m.u = [] has to be commented out.

For an example, compare 'M2/model/initialze\_M2.m' with 'M2/model/initialize\_M2\_input.m'.

In this file you will also see how to incorporate gene expression data.  
First you have to define the parameters that are not associated with gene expression data. This can be done by providing an array with the corresponding parameter numbers.

Thus if parameter number 4 6 and 9 are not associated with gene expression:

m.not\_associated = [ 4 6 9 ];

Then for the parameters that are associated with gene expression, you provide the name of the parameter and give an array of numbers that are associated with gene names like the following:

m.par\_ass\_genes.k\_FC\_cat = [35 24 26];

m.info.genes{35} = 'abcg1';

m.info.genes{24} = 'abcg5';

m.info.genes{26} = 'cyp7a1';

Note that the numbers provided in the array do not matter as long as they correspond to the indexes provided in m.info.genes.

In the folder 'M2', use 'run\_analysis' to start the ADAPT run. If you open the file you will notice some minor changes.

both check\_model\_configuration and perform\_ADAPT now also require 'u' (input) as an argument, 'u' may be empty.   
If you do use input, do:

m = check\_model\_configuration(m, u(:,1,1))   
% if you don't take the slice check\_model\_configuration will fail

[R, m] = perform\_ADAPT(m, u);

In 'multiple\_model\_analysis' in the 'M2' folder you will find a script that gives an example of how you could analyze two separate conditions/datasets using the same model topology.

In the first part of the script, ADAPT is run for the first condition, denoted as 'reference'. Also, a sensitivity analysis is performed on the model and data and results are printed to 'sa\_dataset.xls'.

In the second part of the script, ADAPT is run for the other conditions, denoted as 'target'.

In the third part, a readout of interest is defined, denoted as 'roi', which can either be a (linear combination of) state or flux and the time\_span that you are interested in. This time\_span will determine what time steps will be taken into account to calculate the difference in the readout. It will also determine for what time steps a difference in parameter trajectory predictions between the reference and target model is calculated.

Then, in the fourth part, one parameter trajectory prediction from the reference model will be substituted for the corresponding parameter trajectory prediction from the target model. After this 'perturbation by substitution', the difference in roi is calculated. This is repeated for every parameter. In this way, you can get a sense on how much a parameter trajectory prediction can explain the difference in an observation between the reference and target condition.

Then, in the fifth part, for specified time\_span, the ratio of the parameter value predictions for the target and reference model is calculated, and these ratios are then used to modulate the reference parameter trajectories. In essence, this 'perturbation by modulation' provides you with an alternative way to get a sense on how much a parameter prediction affects differences in your readout of interest.

Results of 'perturbation by substitution' and 'perturbation by modulation' are then stored in 'perturbation.xls'.

In this example you will not find any interesting differences because the same dataset was used for reference and target model.