Delivery date:

31-12-2014

Authors:

Heinz A Preisig
Mandar Thombre
Sigve Karolius

MoDeNa

Deliverable D5.3

Parameter Estimation Framework

Principal investigators:

Heinz A Preisig

Project coordinator:

Heinz A Preisig

heinz.preisig@chemeng.ntnu.no

Contents

1 Modelling concept

MoDeNa is modelling polyurethane over the full range of scales: from quantum to mechanical properties. The multi-scale nature is thus apparent. Models on the higher scale will in general depend on models of the lower scale, thus the models are computed recursively over the scales. ***MoDeNa*** does not work with coupling, meaning the lower scale model is computed on request in each point of its definition space. ***MoDeNa*** uses instead surrogate models that approximate the lower scale model over a subspace of the definition space.

The model hierarchy can be seen as a layered set of models, in which the layers are the scales (Figure 1):

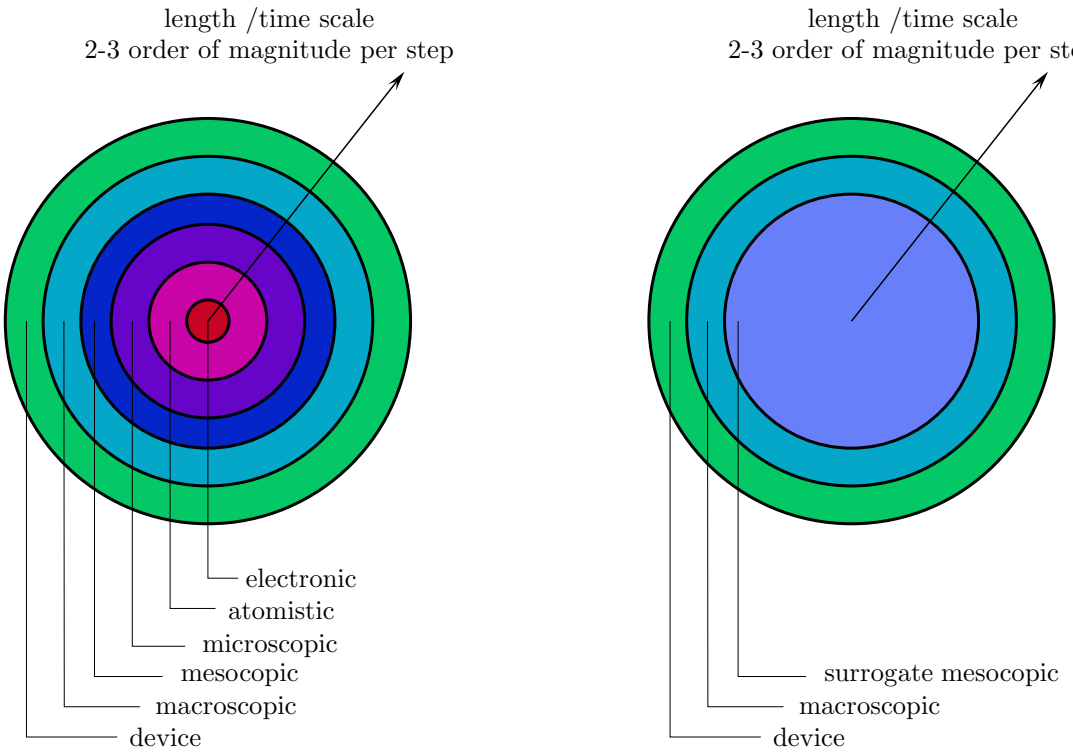

Figure 1: The multi-scale onion and the multi-scale onion with a surrogate core

2 Computation concept

To explain the scheme, we define a model on the scale $i$as an input/output structure[1]:

$$\underline{\mathbf{y}}^i := \mathcal{M}\left(\left[\mathcal{M}_j^{i-1}\right], \underline{\mathbf{u}}^i\right)$$

being a function of a list of models on the next lower level. The input vector $\underline{\mathbf{u}}^i \in \mathbf{U}^i \in \mathbf{U}^i_{\max}$.

The dependency on the lower-level models is often an unacceptable computational burden and one

[1]The symbols are explained in the notation section. The electronic version of the report implements active links for all symbols

replaces these models by surrogate models. Thus the modified computational tasks takes the form:

$$\underline{\mathbf{y}}^{i+1} := \mathcal{M}\left(\left[\hat{\mathcal{M}}_j^i\right], \underline{\mathbf{u}}^{i+1}\right)$$

The substitution was argued based on computational burden, which in computational engineering is likely to dominate. Though it should be noticed that this is also common practice in science and engineering, as philosophically seen, any model is a surrogate of a more complex underlying model.

$$\underline{\hat{\mathbf{y}}}^{i+1} := \hat{\mathcal{M}}\left(\left[\hat{\mathcal{M}}_j^i\right], \underline{\mathbf{u}}^{i+1}\right)$$

The recursion of forming surrogate models is now apparent.

The user of the surrogate model is defining the acceptable error in a definition domain. If the user is moving outside the current definition domain for the surrogate model, the extension of the current definition domain is triggered and a design of experiment for the extended domain is started. This makes the overall scheme event driven. We first define the event formally.

2.1 Event

To each instantiated model a state is being defined, which is *in-range* and *out-of-range*. The latter we define as:

$$\underline{\mathbf{u}}^i \notin \mathbf{U}^i \tag{1}$$

The event of switching to *out-of-range* triggers the surrogate model to be expanded to a larger input range. The mechanism requires an event handler, which operates on identifiers attached to each instantiated model, the current state and the state-change events.

2.2 Surrogate model

The surrogate model $\hat{\mathcal{M}}$ is an approximation of a more complex model $\mathcal{M}$:

$$\hat{\mathcal{M}} \approx \mathcal{M} \tag{2}$$

The structure of the surrogate model is usually given and generating the approximation implies "fitting" the surrogate model to the complex model. The fitting is done for the updated definition space, each time the *out-of-range* event is signalled.

$$\mathbf{U} := \mathbf{U} + \Delta\mathbf{U} \tag{3}$$

The identification requires a measure of "goodness". We define it as a gap.

2.3 Gap measure

The gap between the surrogate model and the model is defined as:

$$\mathbf{g} := \mathbf{g}\left(\underline{\mathbf{y}} - \underline{\hat{\mathbf{y}}}\right) \tag{4}$$

being a generic function of the difference between the output of the complex model and the surrogate model for a given input. So the identification (fitting) of the surrogate model uses this gap definition

to generate a model that satisfies a minimal requirement on the gap between the model response and the response of the surrogate model. Examples for the gap criterion are:

$$\mathbf{g} := \sup \left(\underline{\mathbf{y}} - \underline{\hat{\mathbf{y}}}\right)^2 \tag{5}$$

$$:= \sup \left|\underline{\mathbf{y}} - \underline{\hat{\mathbf{y}}}\right|^2 \tag{6}$$

with sup, the supremum, being the minimum of the maximum values.

2.4 Design of experiments

The model update operation is triggered by an *out-of-range* event. The first action is to extend the range by a given increment of the space though not beyond the maximal input space.

$$\mathbf{U} := \mathbf{U} + \Delta\mathbf{U} \tag{7}$$

Next experiments are designed for the extended domain and the computational experiments are executed. The newly computed input/output data are added to the existing set and the model identification is executed on the complete data set, aiming at identifying a model that meets the error criterion defined by the model that uses the surrogate model.

Defining the acceptable error as $\underline{\epsilon}$ then identification will minimise

$$\mathbf{g}(\cdot) - \underline{\epsilon} \tag{8}$$

which implies that one does not seek the optimum, but good enough so as to minimise the evaluations of the complex model.

3 Issues

- The user of the surrogate model is to define the error criterion $\underline{\epsilon}$. The identification is thus only going to match the error, but does not try to achieve a minimal gap.
- The identification may not be able to meet the goal, for reasons of a too large gap over the defined range. The issue can only be noticed indirectly as the algorithm does not complete. The user of the surrogate model must thus also define a maximal number of model evaluations.
- The identification can also simply take too much time in the case where the model evaluation take too much time. It may thus be advisable to have the user to also define a time limitation.
- The evaluation in some input sub-domain may provide more information than others. One can obtain some guidelines by analysing the sensitivity of the input to the estimated parameters of the surrogate model. So this part of the design of experiments is done purely in the context of the surrogate model.
- Likely approach to the minimisation is to define an empirical surrogate model for the gap and use it in a surface response method for the optimisation - a method that is standard for nonlinear optimisation.

4 Framework for design of experiments and parameter estimation

This section explains the example that has been formulated specifically in order to show the concept of using design of experiments and parameter estimation in a case similar to the ***MoDeNa*** project.

4.1 Approximating the Redlich-Kwong equation of state with a van der Waals type surrogate model

We assume that in a multiscale system, a macromodel in the system (currently undefined) requires pressure data from a micromodel in the system. For the micromodel, we take the example of the Redlich-Kwong equation for the case of ammonia:

$$P = \frac{RT}{(V-b)} - \frac{a}{\sqrt{T}V(V+b)},$$

where a and b are the Redlich-Kwong constants for ammonia.

The constants a an b are calculated using the following equations:

$$a = \frac{0.4275R^2{T_c}^{5/2}}{P_c} \qquad \text{and} \qquad b = \frac{0.08664RT_c}{P_c}$$

where T_c and P_c are the critical constants for ammonia, the values of which are shown in (Table 1). The objective is to approximate this pressure using a van der Waals type equation:

$$\hat{P} = \frac{RT}{(V-b)} - \frac{a'}{V^2},$$

Table 1: Thermodynamic data of ammonia used in the simulation

Parameter	Value/Range	Unit
T	$[420-600]$	K
V	$[0.25, 1.00]$	L mol^{-1}
R	0.082	L atm mol^{-1}K^{-1}
T_c	405.40	L mol^{-1}
P_c	111.52	atm
V_c	0.072	L mol^{-1}

On comparing the Redlich-Kwong and van der Waals equations, it can be seen that the first term on the right hand side of each equation is the same.[2] Hence, to approximate the Redlich-Kwong equation with the van der Waals type equation, we focus only on the second terms on the right hand side of both the equations. The problem thus simplifies to fitting the data obtained from the $\left(\frac{a}{\sqrt{T}V(V+b)}\right)$ term in Redlich-Kwong equation to the $\left(\frac{a'}{V^2}\right)$ term in van der Waals type equation. Thus, the parameter a' in the van der Waals type equation needs to be determined using non linear least squares regression. The values and ranges of the constants and the variables used for the simulation are as shown in (Table 1).

[2] We assume the value of the constant b to be same for both the models because the actual van der Waals value for b is very close to the Redlich-Kwong value for b in case of ammonia.

4.2 The Code

In the code `example.py`, the surrogate model is defined as a function at the beginning. The surrogate model should ideally be called, when required, from the MongoDB database.

The structure of the database is then defined and initialised. The database contains the following information:

1. Information about the DoE: the number of sample points, the obtained values of the temperatures and volumes from the DoE procedure.[3]
2. Information about the micromodel: The values of the pressures calculated at the selected points. Also, the values of the 2nd terms on the RHS of the Redlich-Kwong equation calculated at the selected points (because this is the data that will be fitted).
3. Information about the surrogate model: The parameter a' of the surrogate model obtained after the data fitting and the values of the pressures (at the DoE selected points) calculated using this surrogate model.
4. The error tolerance criteria as set by the macromodel and the validation status depending upon whether this criteria is satisfied. The validation status is 'valid' if the obatained surrogate model satisfies the criteria and 'not valid' otherwise.

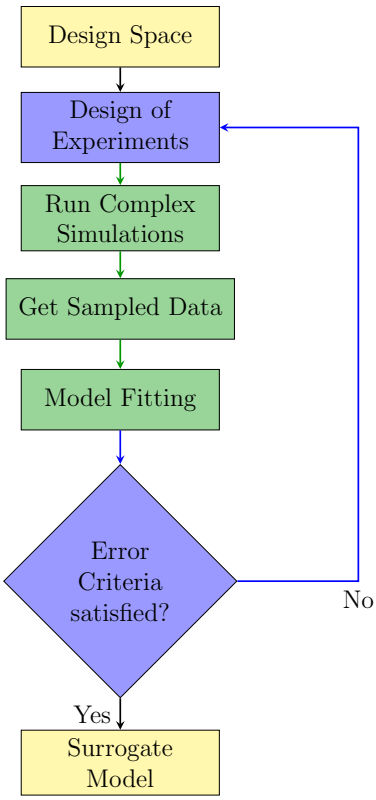

Figure 2: Calculation flowsheet for `example.py`

The first `Firetask` is the MacroModelTask, which connects to the MongoDB database and updates the error tolerance criteria in the database. It then checks the validation status and calls the DoE `Firetask` if it is 'not valid'.

In the DoE `Firetask`, the number of sample points is chosen and updated in the database. The DoE procedure is performed and the obtained values of temperatures and volumes are also updated in the database. The MicroModel `Firetask` is then called.

In the MicroModel `Firetask`, the DoE values of temperatures and volumes are taken from the database. The Redlich-Kwong equation pressures as well as the values of the 2nd terms on RHS of the equation are calculated at these temperatures and volumes and updated in the database. The ForwardMapping `Firetask` is then called.

The ForwardMapping `Firetask` takes the temperature, volume and the '2nd term on the RHS' data from the database and performs the fitting using the surrogate model defined at the top of the code. The obtained parameter a' is updated in the database. This `Firetask` also calculates the pressures using the obatained surrogate model and updates these values in the database. It then calls the BackwardMapping `Firetask`.

The BackwardMapping `Firetask` takes the Redlich-Kwong pressure values as well as the surrogate model pressure values from the database and calculates the error between the two. It compares this error with the error tolerance criteria set by the macromodel, again taken from the database. If the criteria is satisfied, it sets the validation status to 'valid' and prints 'We are done'. Otherwise it calls the DoE `Firetask` and the procedure is repeated again.

[3] The sampling method is implemented on a 2-dimensional design space, the design parameters being temperature and volume. The ranges of temperature and volume are $[420, 600]$ and $[0.25, 1.0]$ respectively.

5 Implementing design of experiments and parameter estimation

The design of experiments and parameter fitting module will both be implemented using a connection to the R programming language. This makes it possible to take advantage of the extensive list of libraries as well as native capabilities that that makes R a popular language with statisticians (R Core Team (2014)).

The current implementation of the module should, however, be considered a "proof of concept". It was designed to be easy to use and modify, and to exemplify a possible strategy for how to approach the implementation of the final and much more elaborate module. The major difference with respect to a future module is the lack of capability to perform model-based design of experiments, which is one of the specifications in the ***MoDeNa*** project.

5.1 Current implementation

In order to keep the example as simple as possible the DoE module (`DesignOfExperiments.py`) works as a servant to a `Firetask` located in the main script (`example.py`). This means that the design of experiments module does not access the database (MongoDB) and thus must be supplied with parameters. The module returns a latin hypercube design of experiments given the desired number of samples (`N`) as well as the range of the input space (**U**). Moreover, the module can also add several samples to the experiment if necessary, but it does not check the quality of the estimated surrogate parameters.

The design is created using the "`lhs`" library to generate the experiments in R and the `pyRserve` package to facilitate communication with Python. The dynamics between the design of experiments module with respect to the main script and database is illustrated in (Figure 3).

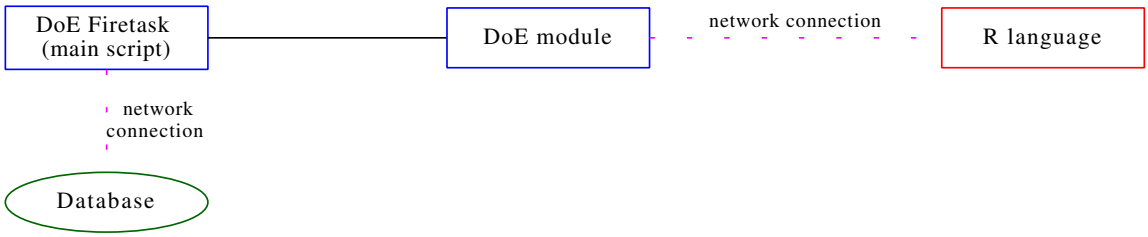

Figure 3: Illustration of the structure between the main Python sctipt, specifically the DoE `Firetask`, that handles communication between the database and the DDoE module which in turns communicates with R.

The figure shows that the module is not communicating with the database and consequently it can be considered "stand alone", meaning that it can be used independently of the ***MoDeNa*** framework.

The independent nature of the module reflects the principle behind the implementation: the module should be easy to modify and intuitive to use. It was designed in an object oriented manner with a main ("parent") class defining the properties of the module (e.g. how to access/retrieve information) and employing "child classes" that have traits that are specific to their functionality, but inherit their properties from the "parent". Consider the following classes implemented in the module:

ParentDesignClass
: This is a "parent class", and the backbone of the module. It defines what happens when any other class is called and how to retrieve and store new information. However, it will never be

seen by anyone that does not look at the code. The reason for this is that it does not know what function to execute in R, this information is specific to the "children".

latinHypercube
This is a "child class" that inherits its properties from the "parent". It also contains the specific function that should be used to create a latin hypercube design of experiment in R. Several "child classes" can be implemented to add more functionality to the module.

The argument that it is intuitive to use is not necessarily intuitive. Considering the bullets below that exemplifies the necessary commands necessary in order to use the module.

- Start a new design of experiments by calling the module as follows:
  `DoE = DesignOfExperiments.latinHypercube(N,i{'D1':[0,1], 'D2':[0,1]})`
- The samples for each input dimension is accessed as shown below.
  `DoE['D1']`
- If the parameter fitting does not produce satisfactory results, add more samples using:
  `DoE.addSamples(M)`
- Alternatively, if it is desirable to scrap the current experiment and start over:
  `DoE.cleanStart()`

5.2 Strategy and challenges for future implementation

The requirement of a model-based approach to design of experiments demands a much more elaborate module described in the previous section. There are mainly two reasons for this: the current module can not expand the range of the design of experiments, nor can it evaluate the quality of the parameters estimated for the surrogate model. It will be necessary to divide the tasks into several modules in order to address the shortcomings. This will in the future be done by separating the tasks into three modules:

New design of experiments
Module responsible for creating a fresh design of experiments in the desired input range.

Parameter fitting
Module responsible for least square regression of the output from the micro model to estimate the parameters of the surrogate. This will also handle the sensitivity analysis and assess wether the quality of the surrogate approximation is sufficient.

Additive design of experiments
Module responsible for expanding the range of the design of experiments or add several points in an attempt to improve the surrogate approximation.

Each module now has a specific purpose and can be made into a `Firetask` that fits into the ***MoDeNa*** framework.

There are several challenges associated with the model-based approach to design of experiments, many of which are listed in (Section 3). The goal will be to implement a parallell approach to design of experiments as illustrated in (Figure 4). This will raise a number of practical challenges, some of which are listed below.

- Storing several design of experiments for the same model
- Allowing a new surrogate model to be added and access the results from previous design of experiments
- Expanding the range of the design of experiments (within pre-defined boundaries)
- Parameter estimation: when/how to give up and settle with a non-optimal set of parameters?
- There are many R libraries, choosing the best for ***MoDeNa*** is essential.

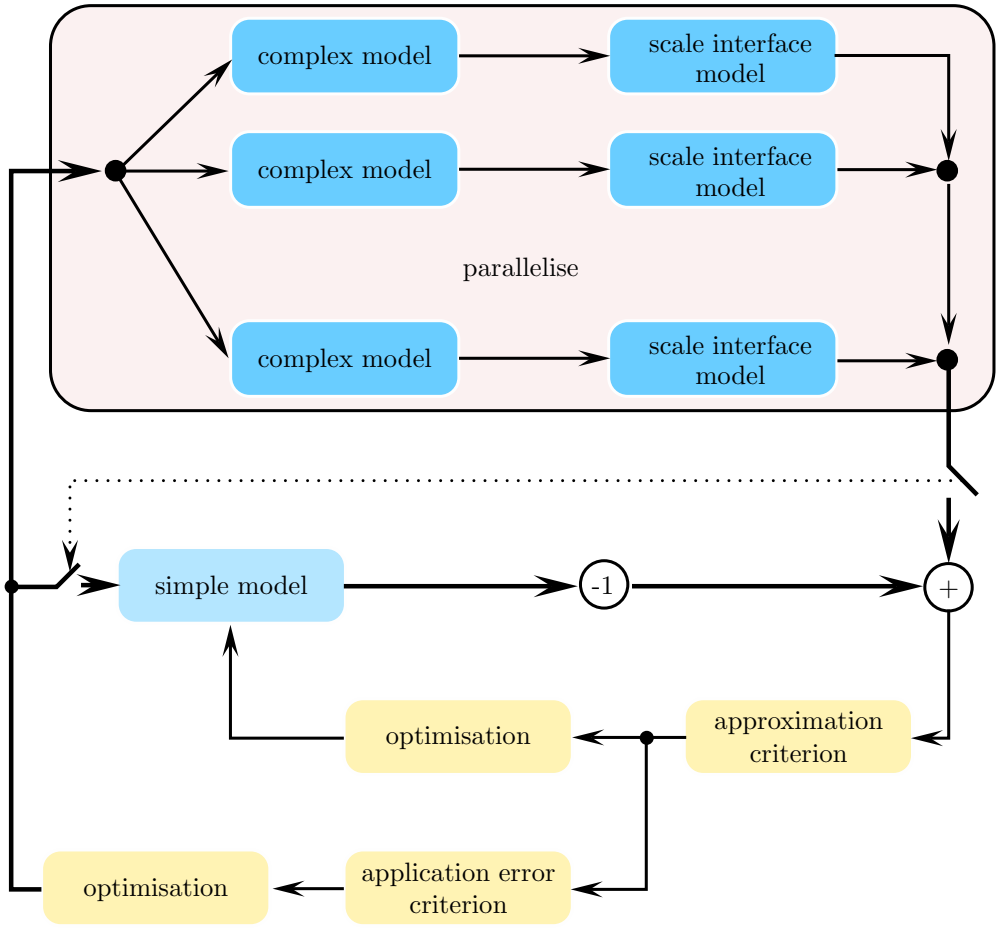

Figure 4: Illustration of a parallell approach to model-based design of experiments

The parallell approach to model-based design of experiments is also outlined in [Franceschini, G. ans Macchietto, S. 2008], albeit for combined computer and physical experiments. The goal is to implement a module as shown in (Figure 4) and generalise it into the ***MoDeNa*** framework in order to comply with the deliverable in the project description. In order to focus on the framework, however, validated R algorithms should be used to perform DoE and parameter estimation.

5.3 Features from the R programing language

There are two strategies for directly communicating with R from Python. The term "communicate" implies that the Python datatype is translated, sent to R for processing and the result is delivered back

after completion.

However, while the goal of the packages is the same, the strategies behind are fundamentally different. As mentioned in (Section 5.1), the current design of experiments uses `pyRserve`. This package facilitates communication between the environments by connecting to R when it is running in server mode. The illustration in (Figure 5) shows that the environments are are running independently.

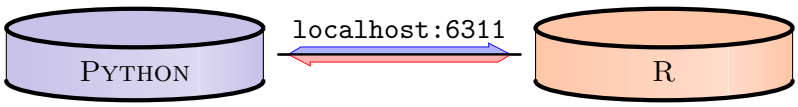

Figure 5: Illustration of the relationship between Python and R in the `pyRserve` package. The text `localhost:6311` specifies that R is running as a local server on port 6311 (default).

The alternative to `pyRserve` is the `rpy2`. The latter is preferred for the ***MoDeNa*** framework based on the experiences made in the current implementation. The reason is that maintaining a server connection will put more strain on the end-user and maintainer of ***MoDeNa***. This is avoided when using `r2py`, whose drawback, seen from a ***MoDeNa*** perspective, is mainly "less attractive syntax".

The strategy behind `rpy2` is to integrate R into Python. This is the reason why it is referred to as "R-in-a-cage", as illustrated in (Figure 6). Note, however, that `rpy2` is not the entire R language bundled into a Python package. It is therefore necessary to have R installed on the computer.

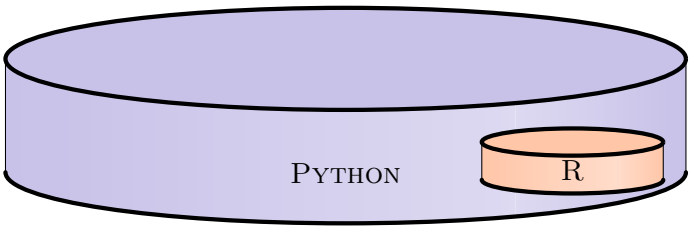

Figure 6: Illustraton of `rpy2` package "R in a cage".

The reason for using R is that it is implemented especially with statistics in mind. The environment natively contain functionalities such as: least square regression, bootstrap algorithms and probability distributions. Moreover, it can easily be expanded with powerful external libraries specifically for the needs of the ***MoDeNa*** project. Several potentially relevant packages have been identified and are shown in (Table 2).

Table 2: Relevant non-standard packages from CRAN (r-project web site)

Package	Contains
lhs	Latin hypercube space filling designs
DiceDesign	Varions techniques for space filling design
sensitivity	Functions for global sensitivity analysis

There are, however, many libraries and it is challenging to predict the needs of all the groups in the consortium. One of the main focuses for the future will therefore be to identify exactly what must be included in the DoE module.

This is where the main incentive for using R becomes clear, it makes it possible to ensure that the

methods are implemented quickly and correctly. Finally, it provides the consortium with flexibility in focusing how to integrate model-based design of experiments into the framework in the best way.

References

Franceschini, G. and Macchietto, S. (2008). Model-based design of experiments for parameter precision: State of the art. *Chemical Engineering Science*, 63(19):4846 – 4872. Model-Based Experimental Analysis.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 5

Nomenclature

$\underline{\epsilon}$ error vector as a minimum specification. 3

$\mathbf{g}$ quality measure, criterion. 2, 3

i scale level. 1, 2

$\mathcal{M}$ mathematical model. 1, 2

$\hat{\mathcal{M}}$ surrogate for mathematical model. 2

$\mathbf{U}$ input space / definition space. 1–3, 6

$\underline{\mathbf{u}}$ input vector. 1, 2

$\underline{\hat{\mathbf{y}}}$ output of surrogate model. 2, 3

$\underline{\mathbf{y}}$ output vector. 1–3

$\underline{\hat{\mathbf{y}}}$ output vector of surrogate model. 2