Prototype implementation of the FORCE BDSS system: Use Case, BDSS architecture and implementation modules

Peter Klein[a,*]

[a]*Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer Platz 1, D-67663 Kaiserslautern, Germany*

Contents

[*]Corresponding author

Email address: `peter.klein@itwm.fraunhofer.de` (Peter Klein)

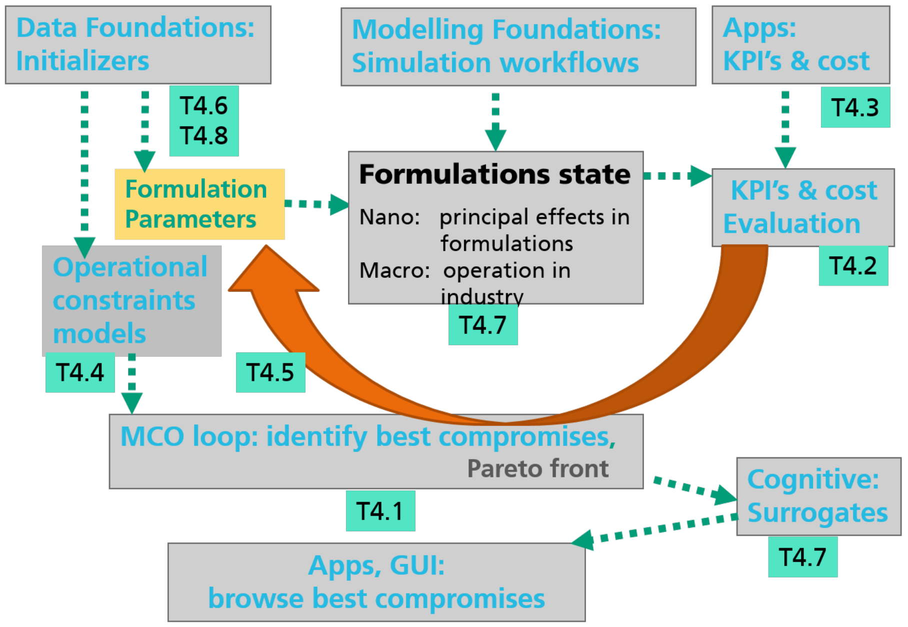
Data Foundations: Initializers
Modelling Foundations: Simulation workflows
Apps: KPI's & cost
T4.6
T4.8
T4.3
Formulation Parameters
Formulations state
Nano: principal effects in formulations
Macro: operation in industry
KPI's & cost Evaluation
T4.2
Operational constraints models
T4.7
T4.4
T4.5
MCO loop: identify best compromises, Pareto front
Cognitive: Surrogates
T4.1
T4.7
Apps, GUI: browse best compromises

Figure 1.1: Coarse level architecture and task mapping of the FORCE WP 4.

1. Introduction

In FORCE, we strive for the development of a general platform supporting Business Decision Support Systems based on materials modeling. This platform will be demonstrated in 3 different application scenarios from the field of chemical formulations. The purpose of this document is to define a first prototype implementation of the FORCE BDSS system, see [1]. Basically, we use the architecture picture from the proposal, figure 1.1, mapped to the tasks of WP4. The idea is to use a toy business case for a chemical process, namely a standard reaction kinetics equation, as a first incarnation of a materials modelling workflow. This model admits an analytical solution, so the testing of all modules of this prototype is fast. Thus, a firm basis for discussions about interfaces and the software design of the final BDSS system will get constructed.

The description of the prototype implementation starts with a hypothetical business case: a toy reaction performed in an idealized reactor. Then, a mapping of the toy system to a simulation workflow will get constructed and the relation of design and objective space (KPI and cost measures) outlined. Together with constraints describing the reactor capabilities, the multi-criteria optimization model and it's solver will be set up. Implementation modules are described in a CRC like style: the responsibilities and collaborations between implementation modules will get identified and a high level description of the data attributes and functionalities of the implementation modules is suggested.

It turns out that the complexity of the modules is very low since all models are given/solved by analytical functions. The focus of the prototype is thus on the interfaces between the implementation modules and on a first attempt to set up and practice an implementation process of the FORCE BDSS to be used by the consortium.

2. The hypothetical business case used in the Prototype

The business case, we want to use in the FORCE prototype implementation as a toy, consists of a hypothetical reaction of two educts, A and B in a liquid:

$$\mathrm{A} + \mathrm{B} \longrightarrow \mathrm{P},$$

where P is the product. In practical applications, no substance is pure and very often side reactions show up. We assume that only one educt, A, contains traces of some non-reactive contaminant, C, and we denote a side product by S. Thus, the above reaction scheme generalizes to

$$\begin{tikzcd}\mathrm{A} + \mathrm{B} \arrow[r, "\mathrm{k_P}"] \arrow[dr, "\mathrm{k_S}"] & \mathrm{P}, \\ & \mathrm{S}\end{tikzcd} \tag{2.1}$$

$\mathrm{k_P}$,$\mathrm{k_S}$ are the reaction constants for product and side reactions. We assume that no other reaction occurs, thus the **side product is distinguished from the product by a lower reaction constant**.

The reaction process runs in a reactor with ideal temperature control and mixing and the reaction preserves the volume (no activities). The process may be stopped by a fast unloading of the reactor and no reaction takes place after unloading. The reaction constants are known to be non-linear functions of the reactor temperature, and the product reaction rate is always higher than the side product reaction rate. Thus, the purity of the product P may be maximized (KPI) by tuning the temperature. But a high temperature leads to a high cost in the temperature control of the reactor. On the other hand, the more contamination C in A is allowed, the cheaper the educt;

but the non-reactive C dilutes the Product P and has thus a negative effect on the quality. We therefore face conflicting goals. In the next paragraph, we will construct a toy kinetic models for the reaction process and a reactor.

3. Kinetic reaction model: simulation workflow and data foundation

3.1. The reactor model

The reactor has a certain volume, V_r, and it is assumed that the loading and unloading of educts from their tanks is much faster than the reactions. The dispersion of the educts is assumed to be ideal throughout the reaction process. The temperature control of the reactor leads to an homogeneous temperature in the reaction volume. The reactor gets loaded with educt B, assumed to be ideal, and educt $\tilde{A}$ which contains, beside educt A, a contamination by C at a known level. The particle densities of all ecducts, ρ_X, and the molecular masses, m_X, are assumed to be known. We denote by $[X]_e$ the volume concentration of a substance X in his feed tank and treat the contaminated educt $\tilde{A}$ as if it had been mixed with substance C. Mixing processes are controlled by a loading volume V_X for each substance. The initial mixture in the reactor is then described by mixture parameters θ_e, θ_m (Volume load fraction). In the $\tilde{A}$ tank, we have the following concentrations:

$$\begin{aligned}
[A]_e =& \rho_A \cdot (1-\theta_e) \\
[C]_e =& \rho_C \cdot \theta_e \\
\theta_e =& \frac{V_C}{V_A + V_C}.
\end{aligned}$$

In the B tank, we have a pure substance at known particle density $\rho_B = [B]_e$.

For the mixture in the reactor, once again controlled by volume loading, we take into account that $\tilde{A}$ gets loaded. This leads to the following volume concentrations in the reactor before the reaction starts:

$$\begin{aligned}
[A]_0 =& \rho_A \cdot (1-\theta_m) \cdot (1-\theta_e) \\
[C]_0 =& \rho_C \cdot (1-\theta_m) \cdot \theta_e \\
[B]_0 =& \rho_B \cdot \theta_m \\
\theta_m =& \frac{V_B}{V_{\tilde{A}} + V_B} \\
[P]_0 =& [S]_0 = 0.
\end{aligned} \tag{3.1}$$

Since we neglect volume effects (activities), the initial concentrations are functions of the mixing parameters and the particle densities of the pure substances only, and product and side product are absent.

The processing cost model takes into account the temperature control and the reaction time τ; the materials cost is expressed using the mixing parameters:

$$
\begin{aligned}
cost_{prod} =& V_r \cdot \tau \cdot (T - 20^\circ\text{C})^2 \cdot W, & (3.2)\\
& W\text{: cost per Kelvin squared and time,} \\
cost_{mat} =& V_r \cdot ((1-\theta_m) \cdot cost_{\tilde{A}} + \theta_m \cdot cost_B), & (3.3)\\
& cost([\tilde{A}]) = (1-\theta_e) * const_A + \frac{const_C}{\theta_e}.
\end{aligned}
$$

$const_A, const_C$ and $cost_B$ are retrieved from pricing informations of suppliers in €/liter. Ideal purity of A has an infinite price. The available contamination levels in $\tilde{A}$ and the realizable temperature range within the reaction volume are assumed to be known:

$$[C]_{min} \le [C]_e \le [C]_{max} \tag{3.4}$$

and

$$T_{min} \le T \le T_{max} \tag{3.5}$$

We assume that loading and un-loading, as well as labour, has negligible costs.

3.2. The kinetic reaction model, related to a simulation workflow

The kinetic model is rather simple. Denoting, as usual, the volume concentration of a substance X by $[X]$, given in $mol/liter$, the kinetics is described by a first order reaction in all components representing the **physics equation**:

$$\begin{aligned}\frac{d[A]}{dt} &= -(k_P + k_S)[A][B] \\ \frac{d[B]}{dt} &= -(k_P + k_S)[A][B] \\ \frac{d[P]}{dt} &= k_P[A][B] \\ \frac{d[S]}{dt} &= k_S[A][B] \\ \frac{d[C]}{dt} &= 0\end{aligned} \tag{3.6}$$

The reaction constants are assumed to be functions of temperature T following the Arrhenius law which represents a **materials relation**:

$$k = \nu e^{-\frac{\Delta H}{RT}} \tag{3.7}$$

$$\nu : \text{Reaction Frequency in } \frac{l}{mol \cdot s}$$

$$\Delta H : \text{molar reaction ethalpy}$$

$$R, T : \text{Gas constant and temperature}$$

The parameters for the materials relation

$$MR = \{\{\nu_P, \Delta H_P\}, \{\nu_S, \Delta H_S\}\} \tag{3.8}$$

are assumed to be known.

We assume that the temperature is actively controlled in a reactor, i.e. kept constant. The reaction may be stopped at a reaction time τ by instantaneous unloading of the reactor. In this case, the kinetic reaction model (3.6) has an analytical solution at constant temperature:

$$\begin{aligned}[A] &= [A]_0 - \alpha(t) \\ [B] &= [B]_0 - \alpha(t) \\ [P] &= [P]_0 + \frac{k_P}{k_P + k_S}\alpha(t) \\ [S] &= [S]_0 + \frac{k_S}{k_P + k_S}\alpha(t) \\ [C] &= [C]_0\end{aligned} \tag{3.9}$$

with $\alpha(t)$ as an analytical function:

$$\alpha(t) = \begin{cases} \frac{[A]_0[B]_0(e^{([B]_0-[A]_0)(k_P+k_S)t}-1)}{[B]_0e^{([B]_0-[A]_0)(k_P+k_S)t}-[A]_0} & : [A]_0 \neq [B]_0 \\ [A]_0(1-\frac{1}{[A]_0(k_P+k_S)t+1}) & : [A]_0 = [B]_0 \end{cases}$$

Since the denominator and the numerator of the branch $[A]_0 \neq [B]_0$ tend to 0 as $[A]_0$ tends to $[B]_0$, we need for numerical stability reasons another representation which explicitly handles this problem. This is accomplished by a series expansion and a factorization of the "zero by zero" division. Here, we just state the result, correct up to 10^{-9} (ppb accuracy):

$$\alpha(t) = \begin{cases} \frac{[A]_0[B]_0(e^{([B]_0-[A]_0)kt}-1)}{[B]_0e^{([B]_0-[A]_0)kt}-[A]_0} & : \epsilon \geq 0.4 \cdot 10^{-2} \\ \frac{1}{2}\left[\frac{[A]_0[B]_0\sum_{n=1}^{4}\frac{([B]_0-[A]_0)^{n-1}(kt)^n}{n!}}{1+[B]_0\sum_{n=1}^{4}\frac{([B]_0-[A]_0)^{n-1}(kt)^n}{n!}} + \frac{[A]_0[B]_0\sum_{n=1}^{4}\frac{([A]_0-[B]_0)^{n-1}(kt)^n}{n!}}{1+[A]_0\sum_{n=1}^{4}\frac{([A]_0-[B]_0)^{n-1}(kt)^n}{n!}}\right] & : \epsilon < 0.4 \cdot 10^{-2} \end{cases} \tag{3.10}$$

$$\epsilon = |([A]_0 - [B]_0)(k_P + k_S)t|, k = k_P + k_S. \tag{3.11}$$

In the limit $[A]_0 \to [B]_0$, the second branch tends (symmetrically in $[A]_0, [B]_0$) to the analytical solution at $[A]_0 = [B]_0$.

We collocate the time dependent **physical state variable**, namely the concentrations, in one vector:

$$X^{mat} = \{[A], [B], [P], [S], [C]\}. \tag{3.12}$$

From mass conservation and by neglecting activities, we know that

$$\sum_x mass_x \cdot [\mathrm{X}] = \rho_{mass}, \text{ constant mass density in the reactor.} \tag{3.13}$$

which, for the system (3.6), boils down to

$$[A] + [B] + 2 \cdot ([P] + [S]) + [C] = const. \tag{3.14}$$

The solution oftThis system needs initial conditions

$$\begin{aligned} X_0^{mat} =& \{[A]_0, [B]_0, [P]_0, [S]_0, [C]_0\} \\ X^{proc} =& \{T, \tau\}, \\ X =& \{X_0^{mat}, X^{proc}\} \quad \text{as the tuneable parameters,} \end{aligned} \tag{3.15}$$

Usually, one is not only interested in one solution. Questions like 'what happens if we use a higher contamination level?' or 'can we reduce the production costs while maintaining our customers specifications?' and others show up naturally. In order to answer these questions, variations (derivatives) of the cost measures and of the concentrations after unloading of the reactor are usually employed. From a purly mathematical standpoint, we may simply differentiate the analytical solutions/approximations given in 3.10. If a numerical solution procedure is used and no analytical expression is available, as a last resort, we may use finite difference schemes. Mathematically, this is possible and used in practice. Thus, an implementation module may support 'mathematical derivatives', like the gradient operator ∇_X.

However, the physical/chemical phase space of X^{mat} is not a plain 5 dimensional real valued space. First of all, concentrations are restricted by $0 \leq [X] \leq \rho_X$, i.e. concentrations may vary between 0 and the pure substance. This may be handled in 'pure math' as well. On the other hand, it is technically not possible to modify one concentration while maintaining all the other ones; this is already reflected for instance in the mixing roles 3.1. The available phase space for the physics variables may, for instance, be represented by an obvious extension of the mixing role 3.1, leading to 4 independent mixing parameters in our case which is one dimension lower than the mathematically feasible 'space of initial consentrations' !!! Thus, physical/chemical meaningful derivatives need to be constructed according to the needs of the questions to be answered!!

3.3. Prototype implementation modules, related to tasks T4.5, T4.6, T4.7 and T4.8

The kinetic model needs to be initialized by the concentrations of the substances, the parameters of the reaction constant's model and the process parameters (reactor). This calls for the following 4 modules:

- a **ReactionKnowledgeGraph** module for the initial concentrations.
- a **MaterialsData** module for the parameters of the Arrhenius reactivities.
- a **ProcessData** module describing the process environment and its parameters.
- a **modelling workflow** module for solving the kinetics (3.6).

The information from the **ReactionKnowledgeGraph** module is rather simple: the product P may be produced by a first order reaction of the two educts, the pure concentrations of the substances A and B should be equal, and an estimate of the reaction time:

Module ReactionKnowledge.mod: of reaction partners.

```
Responsibility: Chemical inference handling based on data in a Knowledge Graph
Collaborators: The informations gathered by domain experts, patents, literature,..
Data: stores a knowledge graph
// return (list of) potential reaction partners for Product P. ;
1 Function $(X, Y)$ $getEducts4Product(P)$
      // look up knowledge graph, for prototype return educts A and B. ;
2     return info $(A, B)$;
// from chemical inference retrieve potential side products ;
3 Function $list$ $getSideProducts(X + Y- > Z)$
      // for prototype, return 1 element $S$ ;
4     return $list$ ;
5 Function $info$ $goodPractice4Reaction(X + Y- > P)$
6     return info $[A]_0 == [B]_0$;
7 Function $time$ $estimateReactionTime4Reaction(A + B- > P)$
8     return estimate;
```

The interface to the **MaterialsData** is trivial: upon request, the reaction enthalpy and the pre-factor is returned.

Database Materials.db: of reaction partners.

```
Responsibility: Stores reaction partner material parameters. Remember: only two
                reactions occur!!
Collaborators: Data space
Data: Information on possible reactions of A and B
1 Function $(MassMolecule)$  $getComponentDat(X)$
      // inspect data base ;
2     return $mass_X$;
3 Function $\rho_X$  $getPureComponentDensity(X)$
      // inspect data base ;
4     return $\rho_X$;
5 Function $\{\nu, \Delta H\}$  $getArrheniusParams(X + Y- > Z)$
      // inspect data base;
6     return $\{\nu, \Delta H\}$;
```

ProcessData are handled similar to the **MaterialsData**. Since the cost measures are sensitive parameters, their gradient calculation according to the tunable parameters is handled in this module as well:

Database Process.db: of reaction partners accessible to processing.

```
Responsibility: Accessor for processing specific parameters, i.e. the reactor.
Collaborators: Excel sheets provide basic parameters.
Data: A reactor description, including the feed by real educts.
1 Function $(cost_{prod}, \nabla_{X_0} cost_{prod})$ $getProductionCost(T, \tau)$
      // parameters for equation (3.2) from excel sheet;
2     return $(cost, \nabla_{X_0} cost)$;
3 Function $(cost_{mat}, \nabla_{X_0} cost_{mat})$ $getCostMaterial(X_0^{mat}, ContaminationLevel)$
      // parameters for equation (3.6) from excel sheet;
4     return $(cost, \nabla_{X_0} cost)$;
5 Function $(ContaminationMin, ContaminationMax)$ $getContaminationRange(eductA)$
      // look up excel sheet;
6     return available contamination range;
7 Function $(TMin, TMax)$ $getTempRange()$
      // look up excel sheet;
8     return available temperature range;
```

The information necessary for the initialization of the kinetic model is now spread over Materials.db, Process.db and ReactionKnowledge.mod. We collocate these information in an Initializer module:

```
Module Initializer.mod: for kinetic model.
  Responsibility: Initializer Strategy for the simulation of reaction kinetics
  Collaborators: Materials.db, Process.db, ReactionKnowledge.mod
  Data: Handle's to collaborators
  Function $X_0$  $getInitDataKinModel(eductA, eductB, productP)$
      // use midpoint for ranges  $(TMin, TMax) = Process.db :: getTempRange()$ ;
      $X :: T = 0.5 \cdot (TMin + TMax)$ ;
      $(ContaminationMin, ContaminationMax)$
                  $= Process.db :: getContaminationRange(A)$ ;
      $X :: [C] = 0.5 \cdot (ContaminationMin + ContaminationMax)$;
      // Product and side product not loaded ;
      $X :: [P] = X :: [S] = 0$ ;
      // $[A] == [B]$, use mass conservation (3.14) ;
      $info = ReactionKnowledge.mod :: goodPractice4Reaction(A + B -> P)$ ;
      $info -> (X :: [A] = X :: [B] = 0.5 \cdot (100\% - X :: [C]))$ ;
      // reaction time from chemical inference ;
      $X :: \tau = ReactionKnowledge.mod :: estimateReactionTime4Reaction(A + B -> P)$ ;
      return X;
  Function $M$  $getMaterialRelationData(A + B \to P)$
      $S = ReactionKnowledge.mod :: getSideProduct(A + B \to P) :: listentry$ ;
      $M :: \{\nu_P, \Delta H_P\} = Materials.db :: getArrheniusParams(A + B \to P)$ ;
      $M :: \{\nu_S, \Delta H_S\} = Materials.db :: getArrheniusParams(A + B \to S)$ ;
      return M;
```

Now, the Simulation workflow has only one dependency to Initializer.mod, in accord to figure 1.1. At constant temperature, the kinetic model has an analytical solution. The prototype **Sim-Workflow** module module implements the analytical solutions in order to return the physical state variable associated with the solution of the kinetic model. This module acts as a persistency layer for simulations already performed (another design option could be the introduction of an additional data base). The idea is that once the simulation of a materials model takes considerable computing resources, then results might be cached and reused (including interpolations if appropriate), thus saving a lot of computing time. Beside the physical state variable (3.12) the gradient in the space of tunable parameters (3.15) is calculated.

Module ReactionKinetics.mod: simulates reaction kinetics.

Responsibility: Simulation of reaction kinetics

Collaborators: Initializer.mod

Data: Persistency layer for storage of simulations already performed.

```
Function  $(X^{mat}, \nabla_{X_0} X^{mat})$  $runDefault(eductA, eductB, productP)$
    // Use Initializer.mod and run the simulation;
    $X = Initializer.mod :: getInitDataKinModel(A, B, P)$;
    $M = Initializer.mod :: getMaterialRelationData(A, B)$;
    return $self :: run(X, M)$;
Function  $(X^{mat}, \nabla_{X_0} X^{mat})$  $run(X_0, M)$
    if  $(X_0, M)$ simulation cached then
        return cached result;
    else
        // implement solver of kinetic model (3.6);
        calculate $X^{mat}(X :: X_{proc} :: \tau)$ according to (3.10);
        calculate $\nabla_{X_0} X^{mat}$ ;
        cache $\{\{X_0, M\}, \{X^{mat}, \nabla_{X_0} X^{mat}\}\}$ ;
        return $(X^{mat}, \nabla_X X^{mat})$ ;
    end
```

4. Multi criteria optimization in FORCE mapped to the prototype

4.1. General nonsens: MCO for continuous problems

In general, an MCO model has three ingredients:

- An objective space O. Each component of a vector from O represents one criterion to be optimized. By mathematical convention, we always seek for their minimization (by chancing sing on criteria to be maximized).
- A design space D spanning the parameter space mapped to objectives.
- A set of constraints acting in the design space and/or in the objective space, usually given as equalities and inequalities (both linear and non-linear)

The design variables associated with the MCO problem in the prototype are continuous and the objectives are differentiable. Furthermore, the objectives in the prototype are functions of control variables (reaction time, temperature) and of a dynamical state variable. Thus, in the prototype we deal with a multi-objective control problem. In general, these type of problems are formulated mathematically as

$$\min_{y \in D} \{O_1(X(y), y), ..., O_n(X(y), y)\} \tag{4.1}$$

subject to

$\{c_i(y) = 0\}$: design space equality constraints

$\{C_i(y) \leq 0\}$: design space in-equality constraints

$\{g_i(O) = 0\}$: objective space equality constraints

$\{G_i O \leq 0\}$: objective space in-equality constraints

$\{F_i(t, X, \dot{X}) = 0\}$: state constraints, implicit rep. of ODE/PDE

The notation *min* is commonly used here as a short hand for Pareto optimality. Since usually the objectives represent conflicting goals, they may not be minimized simultaneously and we seek for best possible compromises.

Most solvers of MCO problems of this type, so called gradient based solvers, use a common strategy:

- scalarization: use a scalar product of a weight vector with the vector of objectives, thus mapping the MCO to a single criterion optimization problem.
- solve the single criterion control problem using non-linear programming (see [2] for details), the theory behind single criteria optimization problems resides on the Karush-Kuhn-Tucker theorems, see [3]:
  - solve ODE for a given design space point.
  - calculate objectives and derivatives.
  - advice new candidate design space point.
  - stop once the KKT conditions are fulfilled.

  This leads to one candidate Pareto point (usually locally Pareto).
- use a strategy to modify the weight vector in order to find the next candidate Pareto point.
- If the MCO problem is not globally convex, we need to apply a filter mechanism in order to get rid of dominated points not on the global Pareto front.
- once a good enough approximation of the Pareto front is obtained: stop.

An impression on how to classify MCO problems according to the nature of the objectives and of the constraints can be found in the wikipedia articles cited above. Even within one class of MCO problems, the choice of the weight strategy and of the KKT-solver is not unique and needs to get adapted to the specific MCO problem.

4.2. Design space, constraints and objectives for the prototype MCO problem

The model ingredients will now get constructed for the prototype business case; in the language of the EMMC, this construction is a **translation process**.

It is obvious that here we have some freedom in constructing an MCO model. For instance, we may use as design space the space of variables closely connected to the mathematical formulation given as tunable parameters, equation (3.15), subject to mass conservation (3.14) and mixing role constraints (3.1). This modelling would serve the needs of an engineer interested in 'understanding'. In this prototype, we take a somewhat different point of view: the design space shall closely reflect the tunable parameters of the process. We use as design space the process parameters and describe the mixing process by appropriate volume fractions. We thus have a 4 dimensional design space with obvious constraints:

$$D = \{(V_{\tilde{A}}, [C]_e, T, \tau) \| 0 \le V_{\tilde{A}} \le V_r, [C]_{min} \le [C]_e \le [C]_{max}, T_{min} \le T \le T_{max}, 0 \le \tau\} \tag{4.2}$$

The volume load related to substance B is then given by $V_r = V_{\tilde{A}} + V_B$. The initial concentrations, as functions of the design variables, now read:

$$\begin{aligned}
[A]_0 &= [A]_e \cdot \frac{V_{\tilde{A}}}{V_r} = \rho_A \cdot (1 - \frac{[C]_e}{\rho_C}) \cdot \frac{V_{\tilde{A}}}{V_r} \\
[B]_0 &= \rho_B \cdot \frac{V_r - V_{\tilde{A}}}{V_r} \\
[P]_0 &= 0 \\
[S]_0 &= 0 \\
[C]_0 &= [C]_e \cdot \frac{V_{\tilde{A}}}{V_r}
\end{aligned} \tag{4.3}$$

This modelling closely reflects the needs of an operator running the reactor.

Now, we construct the mapping from the design space to the objective space. In natural language we have 3 objectives:

- maximize the product P, which is equivalent to a minimization of product impurities$[I]$.
- minimize the production cost $cost_{prod}$.
- minimize the materials cost $cost_{mat}$.

These product impurity is naturally modelled as a functions of the physical state variables at the un-loading time and implicitly as a function of temperature (via the reaction constants):

$$\begin{aligned}
[I] =& [S] + [C]_0 + res([A]) + res([B]) \\
& res(X)\text{: residual concentrations of X=A,B at final reaction time,}
\end{aligned} \tag{4.4}$$

We collocate the KPI (4.4) and the cost measures (3.2), (3.3) in one 3 dimensional vector O. The mapping of the design space to these vector of objectives uses the analytical expressions obtained above, re-written as functions of the design variables in $y \in D$. For the physical state variable X^{mat}, this is done by using the initial conditions (4.3) and the reaction constants, as functions of temperature T, from the Arrhenius equation (3.7). Similar algebraic manipulations of the mixing parameters lead to cost measures as function of variables from the design space. Thus, the 3 objectives are given by

$$
\begin{aligned}
O_1 =& [I](X(y)) \\
O_2 =& cost_{prod}(X(y)) \qquad (4.5) \\
O_3 =& cost_{mat}(X(y)) \qquad (4.6)
\end{aligned}
$$

and derivatives are readily constructed analytically using the chain role based on the analytical solution for the time depended state variable given in equation (3.10).

The MCO problem, formulated as a control problem, now reads:

$$
\begin{aligned}
&\min_y \{O_1(X(y)), ..., O_3(X(y))\} \\
&\text{subject to} \\
&y : T \leq T_{max} \text{from } (3.5) \\
&y : T \geq T_{min} \\
&y : [C]_e \leq [\mathrm{C}]_{max} \text{from } (3.4) \\
&y : [C]_e \geq [\mathrm{C}]_{min} \\
&\text{ODE}(3.6) \text{ with materials relation and initial conditions as functions of y} \\
&\tau \leq \tau_{max}
\end{aligned}
$$

We restrict the reaction time by a τ_{max} for computational reasons: the contamination of the final product may be minimized at infinite reaction time only, which for a simulator means infinite computing time (similar to convergence criteria of linear algebra solvers). We see, that up to the ODE constraint, all constraints are linear in-equalities. (*Note: if we use the 'mathematical' formulation discussed above, then the constraints need to reflect the mixing roles as a non-linear equality.*)

The formulation as a control problem, however, has some technical difficulties related to the MCO solver implementations to be used in the prototype, namely that the MCO solver implementations 'want' to solve the ODE internally using their own solver and estimates the gradient informations. In a more general situation we use a modelling workflow and this needs to get integrated in a deep manner into the MCO solvers (too much for a prototype!!). Since we have an analytical solution at hand, we use in this prototype another MCO modelling which uses these analytical solutions and thus turning the control problem to a standard MCO problem.

$$
\begin{aligned}
&\min_y \{O_1(X(y)), ..., O_3(X(y))\} \\
&\text{subject to} \\
&y : T \leq T_{max} \text{from } (3.5) \\
&y : T \geq T_{min} \\
&y : [C]_e \leq [\mathrm{C}]_{max} \text{from } (3.4) \\
&y : [C]_e \geq [\mathrm{C}]_{min} \\
&y : \tau \leq \tau_{max}
\end{aligned}
\tag{4.7}
$$

the functions $X(y)$ are then the solutions (3.10) of the kinetic model (3.6) with initial conditions (4.3).

5. MCO implementation modules for the prototype

5.1. MCO solver and associated wrappers for MCO problem set-up

The description of the implementation modules used in the FORCE prototype starts from the central result: a point-wise approximation of the Pareto front.

Database ParetoProcess.db: represents the solution of the BDSS system

```
Responsibility: Handles a point-wise representation of the Pareto surface and -front.
   // return the current representation of the Pareto solution.;
1  getParetoPointContainer(A + B− > P) ;
   // intput is a Pareto optimal desing point and corresponding objectives;
2  insert({ y , O(y) }) ;
```

One interface will be used by the MCO module, the other one by a tool which allows to explore the Pareto surface.

In FORCE, we decided to define one interface wrapper around existing MCO solver implementations. This approach makes the MCO solver exchangeable. While analysing MCO solver implementations, a common style for interacting with design spaces, objectives and constraints can be identified. This allows for a wrapper design in tandem of 4 wrappers around design space, objectives, constraints and MCO solvers. First, we describe the MCO wrapper which internally calls all the other wrappers, and extract from there the interfaces to the 3 other wrapper modules.

Module MCOwrapper.mod: solves the MCO problem.

Responsibility: Wrapper interface to existing MCO solvers

Collaborators: uses internally an MCO solver implementation MCOimpl, Objectives.mod and Constraints.mod

```
   // Initialization ;
1  Function Init ( MCOsetup objects: Objectives and Constraints )
       // Init() is specific to the MCOimpl used!! ;
2      convert MCOsetup in FORCE to MCO setup in MCOimpl;
   // Basic MCO Loop in MCOimpl, included for clarity here !!! ;
3  Function $MCOimpl : solve()$
4      while Pareto solutions not dense enough do
           // Scalarization:  uses a strategy implemented in ACADO or DAKOTA ;
5          $weight_3 = MCOimpl : ScalStrat : getWeights()$ ;
           // construct new objective function evaluation procedure ;
6          $newObjecive = weight_3 \cdot MCOimpl : objectiveInterface()$;
           // Solve scalar problem, inner loop, KKT solver 'knows' constraints ;
7          $MCOimpl : KKT_solver(newObjective)$;
           // store Pareto optimal result  ;
8          $MCOimpl : storeCurrentResult()$;
9      end
   // KKT solver in MCOimpl, included for clarity here !!! ;
10 Function $MCOimpl : KKT_solver()$
11     $y = MCOimpl : InitValueDesignSpace()$;
12     while KKT optimality conditions not reached do
13         $objVal = singleObjective(y)$; $gradobj = \nabla_y of singleobjective$;
            $suggestnewybasedongradobj(linesearchforinstance)$;
14     end
15 Function $solve()$
16     $MCOimpl : solve()$;
17     ParetoProcess.db $= ParetoFilter(MCOimpl : results)$. ;
```

From the MCO module description, MCOwrapper.mod, we know that a MCOsetup is necessary for the MCO solver wrapper. Obviously, the MCOsetup is closely connected to a general MCO problem described in (4.1). We start with a KPI wrapper

Module KPI.mod: for reaction kinetics.

Responsibility: Calculates the KPI's and mathematical gradients

Collaborators: ReactionKinetics.mod, Initializer.mod

Data: Stores materials realtion data M by accessing the Initializer.mod

```
// Prototype implementation:  $[I]$ and derivatives  ;
1 Function  $([I], \nabla_X [I])$   $KPI(X)$
2     $(X^{mat}, \nabla_X X^{mat}) = ReactionKinetics.mod :: run(X, M)$ ;
3     calculate $[I]$ by equation (4.4) ;
4     calculate $\nabla_X [I]$ by using $\nabla_X X^{mat}$ and the chain rule ;
5     return $([I], \nabla_X [I])$;
```

In this module, for flexibility reasons, we use the math gradient. As we have seen above, the construction of an MCO problem is not straight forward and needs to get adapted to the persona using the final BDSS system.

The KPI and the cost measures are collocated in the objective module. We use here directly the design space of the prototype MCO system given in (4.7) at the interface, which is simply an array of dimension 4:

Module Objectives.mod: used in prototype

Responsibility: calculates objectives and their gradients in the design space

Collaborators: KPI.mod, Process.db

```
1 Function $(O, \nabla_y O) objCalc(y \in D, (4.2))$
2     $X = transform(y)$ by equation (4.3) ;
3     $(O, \nabla_X O) :: I = KPI.mod :: KPI(X)$ ;
4     $(O, \nabla_X O) :: prod_cost = Process.db :: calcProdCost(X)$ ;
5     $(O, \nabla_X O) :: mat_cost = Process.db :: calcMatCost(X)$ ;
6     $(O, \nabla_y O)$ calculated from "$(O, \nabla_X O)$ and the chain role ;
7     $return(O, \nabla_y O)$
```

All constraints are handled in a constraints module. In the prototype MCO (4.7), we use linear constraints in each individual dimension of the design space. *(However, much more general constraints are possible, see* (4.1) *).* A special role is played by the constraint on the reaction time: this constraint is for numerical stability/feasibility reasons and thus needs to be specified by a Translator with the help of the ReactionKnowledge.mod.

Module Constraints.mod: for reaction kinetics.

```
Responsibility: handles the linear constraints of the prototype MCO (4.7)
Collaborators: Process.db, ReactionKnowledge.mod
1 Function $range_info$  $getLinearConstraints(DesignSpaceDim)$
      // assumes a 0 based counting;
2 Function $(ContaminationMin, ContaminationMax)$  $getContaminationRange(eductA)$
      // look up data space;
3     return available contamination range;
4 Function $(TMin, TMax)$  $getTempRange()$
      // look up data space;
5     return available temperature range;
  // Numerical feasibility ;
6 Function $\tau_max$  $maxReactionTime()$
7     $\tau = ReactionKnowledge.mod :: estimateReactionTime4Reaction(A + B \to P)$ ;
       // Translators guess;
8     $\tau = \tau \cdot 10$ ;
9     return $\tau$;
```

The main driver for the prototype is now rather short:

Procedure Main Returns the decision vector D and the objective vector O of a Pareto optimal solution.

```
1 Function Main()
2     { MCO description } ;
3     Constraints.mod::init() ;
4     Objectives.mod::init() ;
5     MCOwrapper.mod::init() ;
6     MCOwrapper.mod::solve() ;
```

5.3. Approximation of the Pareto front and cognitive surrogates, T4.7

Using cognitive tools, the ParetoProcess.db will be extended towards continuous interpolation between its entries. The model obtained this way will used in a real time explorer of Pareto optimal

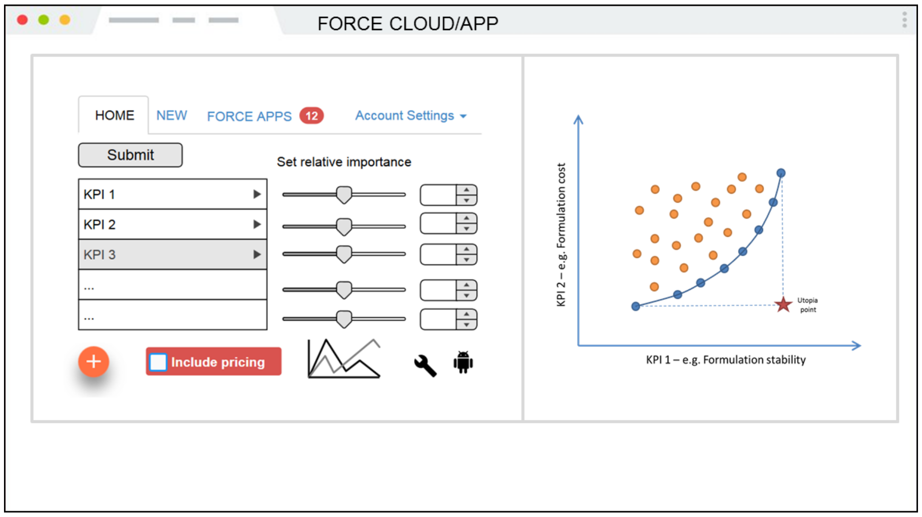

Figure 5.1: Mocup of an interactive Pareto surface and - front explorer from the FORCE DoA.

product and production schemes. This explorer may be implemented similar to the mocup given already in the FORCE description of action, see figure 5.1.

Within the FORCE consortium, we already have access to such explorers, for instance by ModeFrontier (ES).

6. Acknowledgement

The major part of his work was done in the FORCE project, funded by the European Commission within the H2020-LEIT-NMBP program, Grant Agreement No. 721027.

7. References

[1] FORCE consortium, project weg page, https://www.the-force-project.eu/, under construction (2017).

[2] Unkown, Wikipedia, https://en.wikipedia.org/wiki/Nonlinear_programming, Nonlinear Programming (2017).

[3] Unkown, Wikipedia, https://en.wikipedia.org/wiki/Karush-Kuhn-Tucker_conditions, Karush-Kuhn-Tucker conditions (2017).