Towards a Framework for Automatic Event Detection for Car Crash Simulations

Dr. D. Steffes-lai, M. Pathare
(Fraunhofer Institute for Algorithms and Scientific Computing SCAI, Germany)

Prof. Dr. J. Garcke
(Fraunhofer SCAI, Fraunhofer Center for Machine Learning, and Universität Bonn, Germany)

Abstract

In the virtual product development of a car, numerous design changes are applied and analysed until the final model satisfies given design criteria. We focus on crashworthiness requirements, where this procedure usually results in large development trees with many design changes and corresponding simulation results. Following a path in the obtained development tree, the differences from one simulation to the next consist of one or several design variations, which result in numerous changes in the crash behaviour. These differences are mainly determined by local changes in a subset of the car parts.

During the development process each of the design variations has to be analyzed and its influences on the crash results have to be compared and evaluated. To simplify and structure this process, we developed a framework to easily analyze the impact of design variations, including a structured data representation. First, we analyse and store the design variations from one model to the next. Second, in order to detect events, such as anomalies or unusual variations in mesh quantities such as deformations or plastic strain, we analyse and store the changes between the two obtained simulation results. Moreover, we developed a tool to automatically highlight the most relevant parts together with the local areas of high deviations between the two simulation runs. The comparison can be based on arbitrary node- and element data functions, e.g. displacements, plastic strains, or thicknesses.

The presented framework is an important step towards automatic design and event detection in an overall simulation data analysis workflow. It allows to systematically analyze a design development tree to detect interesting deviations in the crash behaviour caused by design changes and to store these observations in a structured data representation for further analysis by for example artificial intelligence approaches.

We demonstrate the framework on a frontal car crash example.

1. Challenges in Crashworthiness Development Process

Nowadays, numerical simulations of full vehicles are an essential factor in car development in order to satisfy challenging legal requirements and customer needs. During the virtual product development, numerous design changes are applied and analysed until the final model satisfies given design criteria (Meywerk 2007). We focus on crashworthiness requirements, where this procedure usually results in large development trees with many design changes and corresponding simulation results. Following a path in the obtained development tree, the differences from one simulation to the next consist of one or several design variations, which result in numerous changes in the crash behaviour. These differences are mainly determined by local changes in a subset of the car parts. During the development process each of the design variations has to be analyzed and its influences on the crash results have to be compared and evaluated. A manual comparison of the results is cumbersome and reaches its limits on the huge amount of data available (Kracker et al. 2020).

In recent years machine learning algorithms have been investigated for crash analysis, in particular using dimensionality reduction to simultaneously analyse large simulation results on the mesh (Bohn et. al. 2013, Iza-Teran et al. 2014, Garcke & Iza-Teran 2017, Kracker et. al. 2020). Rule mining algorithms were investigated to provide an engineer a few reasonable and interesting rules, but the approach was seen to need a couple of hundred simulations for reliable rules (Diez et. al. 2017). Further progress on global analysis of several simulation results to identify clusters with similar simulation behaviour (Steffes-lai et. al. 2021) was presented recently.

Overall, there is a lack of simple to use procedures to automatically categorize design measures together with their impact on simulation results. Towards that goal, an automatic local event detection starting with the consideration of only two design variants is needed as a first step.

To simplify and structure this process, we introduce a framework to easily analyse the impact of design variations, including a structured data representation, as an important step in the simulation data analysis workflow.

2. Framework for Automatic Local Event Detection

During the development process each of the design variations has to be analyzed and its influences on the crash results have to be compared and evaluated. To overcome the burden of manually determining the differences and to structure this process, we developed a framework to easily analyze the impact of design variations, including a structured data representation. The framework allows to systematically analyze a design tree to identify design measures applied in the current design and to detect deviations caused by these

design changes in comparison to a reference simulation. The reference simulation, shortly denoted by reference in the following, can either be the root of the design tree, called global reference, or the direct predecessor of the current design in the overall design tree. Thus, our approach provides insight starting with only two simulations, without the necessity of setting up a simulation data base.

In detail, first, we analyse and store the design variations, called design measures, from the current design to the reference. This step archives all applied design measures and categorizes it in a structured way. Details are given in Section 3. Second, in order to detect events, such as anomalies or unusual variations in mesh quantities, we analyse and store the changes between the two obtained simulation results. Details on the concept for detecting local events are given in Section 4. Moreover, we developed a tool to automatically highlight the most relevant parts together with the local areas of high deviations between the two simulation runs. The comparison can be based on arbitrary node- and element data functions, e.g. displacements, plastic strains, or thicknesses. Furthermore, reports in pdf format can be generated automatically for both the steps, based on the structured result data representation.

3. Design Measures

The first step of the framework deals with a comparison of two input configurations for a numerical simulation. Thus, it consists of the identification of the design measures realized through model adaptation. Design measures could be changes in geometry as well as in attributes such as material property and part thickness. The identification of design measures is based on parts and their discretization (mesh), and thus independent of the parts meta data, such as their identifier or name. Examples of design measures (groups) are changes in geometry, multiparts, that is several parts are merged into one part, changes in spotwelds and rigid body elements, material ID and thickness changes. All the identified design measures are stored in a structured format, namely in JSON files. Different files are generated for different categories, e.g. for changes in part, material, thickness etc. For each design measure group a JSON property exists. The values contain the part-ID and the part-name used in the two models, and additional information like the changed thickness value etc. An example is given in Section 5. More details on the identification of design measures can be found in Garcke et. al. 2017. We implemented this design measure identification process, together with a structured representation and visualization in the Animator[(1)] plug-in ModelCompare[(2)].

[(1)] Animator4 from GNS mbH, gns-mbh.com/animator.html

[(2)] https://www.scai.fraunhofer.de/en/business-research-areas/numerical-data-driven-prediction/products/modelcompare.html

As already stated, the correspondence between the two models is established on the parts discretization. That is, our proposed framework supports the comparison of two similarly discretized finite element models, even with changed or combined parts. The part-ID of the current model corresponding to the part-IDs of the reference is identified. This information is then also used in the second step analysing the obtained simulation results.

4. Local Event Detection

The second step of the approach deals with the comparison of the two obtained simulation results based on the design measures applied, using the full output data, namely displacement and functions on meshes. The approach analyses functions on meshes rather than scalar data or sensor / curve data. More precisely, the impact of design changes can be analysed in terms of data functions either on the nodes (such as displacements or nodal mass) or elements (such as plastic strains, stresses, or failed elements) depending on the analysis objective. For analyzing displacements, rigid body motion can be extracted by setting corresponding anchor points. Thereby, detailed insight in local influences can be evaluated with respect to a certain design change. Several distance measures are investigated to focus on global or local influences. As basic distance measures the maximum norm, the L2 norm as well as the L1 norm are evaluated. To allow the comparison of the difference in a part in relation to all other parts, the metric values are normalized to [0,1]. The maximum norm detects events with emphasis on parts with very high deviations on single nodes / elements between the two models. In contrast the L1 metric sums over the deviations (in elements or nodes) of a part. Thus, applying this metric, events are detected which express in deviations on larger areas of the part selected, that is it puts more emphasis on the entire view of parts. The L2 metric is kind of mixture of both aspects, that is due to the sum of squared deviations single very high deviations are weighted stronger. All differences are stored node- / elementwise for later evaluation.

The comparison results, that is the local events found, are stored partwise in structured JSON files supporting further post processing. In particular, for each part selected, the part-ID, part-name, metricvalue, as well as minimum and maximum deviation in the part is stored.

For interactive exploration, we developed the tool SimCompare[(3)] as an Animator plug-in which enables the visualization of node- and elementwise deviations on both models. This simplifies the detection of local hotspots. The whole vehicle or the relevant parts are shown to get an overview of design change impacts.

[(3)] https://www.scai.fraunhofer.de/en/business-research-areas/numerical-data-driven-prediction/products/simcompare.html

5. Use Case: Toyota Yaris Frontal Crash

We demonstrate the framework on a frontal car crash example. In detail we analyse an open source FE model of the Toyota Yaris[4] from 2010. A EURO NCAP front crash scenario with a speed of 56 km/h against a rigid barrier is simulated in the presented example. Some detail numbers of the Yaris model analysed in this use case are given in Table 1.

Table 1: General model information of Toyota Yaris model.

No. of Nodes	998116
No. of Shells	950343
No. of Shell Parts	732
Total no. of Elements	974171
Total no. of Parts	775

Starting with the reference design, we slightly modify the position of the bumper (PID 2000132). In Figure 1 we show the reference and the current design at state 18, runtime 85ms from total 120ms, that is before bounceback starts. The figure shows the displacements in x direction that is the direction of movement in the load case investigated. The rigid body motion is extracted using three follower points.

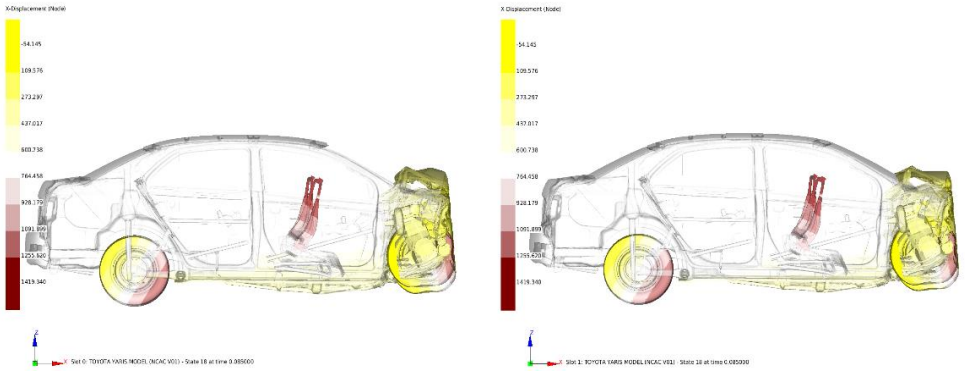

Figure 1: Comparison of the x-displacements of reference (left) and current variant (right) in state 18. The rotation of the bumper lead to a different crash behaviour of the hood.

The figure shows that the design measure applied, namely the slight rotation of the bumper, clearly leads to a different folding behaviour of the hood compared to the reference. However, it is not obvious how this change further affects other parts in the car.

[4] http://www.ncac.gwu.edu/vml/models.html

It is a cumbersome task for the engineer to analyse the simulation results manually and compare functions of interest carefully on relevant parts depending on the analysis purpose.

In the remaining part of this section, we apply our proposed framework to this use case. In a first step, we want to identify automatically the design measures we applied to the current design.

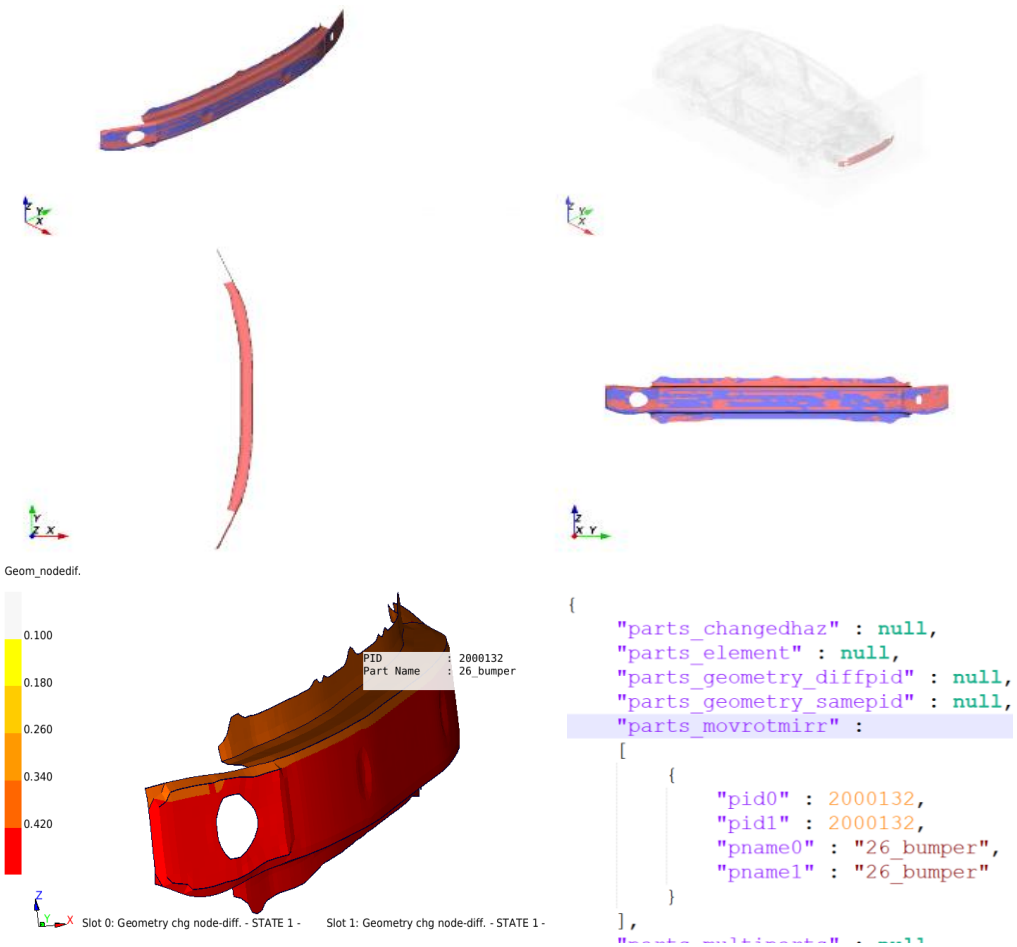

Figure 2: Identified design measure by SCAI tool ModelCompare as GNS Animator plug-in together with an extract of one of the corresponding JSON files: small rotation in the bumper PID 2000132, above from automatically generated ModelCompare PDF-Report, below results visualized in GNS Animator.

With our approach we could exactly identify one design measure, that is more precisely a slight rotation in the bumper (part ID 2000132) shown in Figure 2. This design measure is categorized in the group moved / rotated / mirrored parts, and listed in the appropriate JSON property with part-ID and part name, see Figure 2.

In a second step, we analyse the impact of this single design change on the frontal crash behaviour of the Yaris model. Therefore, we apply our proposed local event detection method on these two design variants. We consider the displacements in x direction here, while the analysis of max. plastic strain gives similar results in this scenario.

The workflow of the local event detection is as follows. First, the differences in x-displacements between the two simulations are computed for every node. These deviations in the displacement per part are then ranked using one of the metrics described in Section 4. This allows to filter for the (by the design change) most affected parts with respect to deviations in the x-displacements. If the value for a part is greater than a chosen threshold, that part is filtered out as belonging to the most affected ones. All the most affected parts are stored in a JSON file together with their metric values. In Figure 3 the most affected parts using a threshold of 0.101 and L1 norm are visualized in green color gradient from high metric values to lower metric values in descending order. We observe that only parts in the front part of the car up to the firewall are highly affected by the applied design change in terms of deviations in the x-displacement.

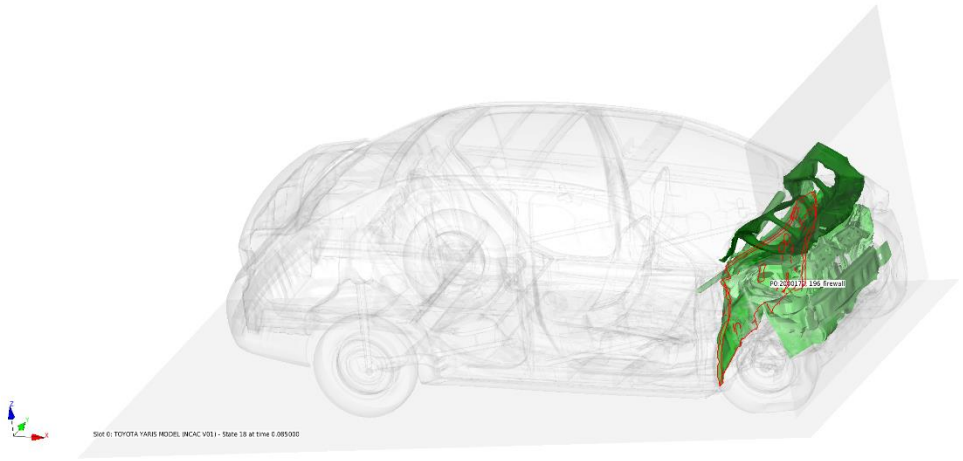

Figure 3: Most affected parts filtered out by SimCompare using threshold 0.101 and L1 norm. Color coding is from dark green, that is highly affected parts (high metric values) to light green (smaller metric values, but still above the threshold). Only parts in the front part of the car up to the firewall are highly affected by the applied design measure in terms of deviations in x-displacement.

We further narrow down the selection of parts by applying a higher threshold. Figure 4 shows the resulting filtering using a threshold of 0.227. The inner hood and the firewall are now highlighted as the most affected parts. On the right hand of the figure, the distribution of the deviation in x-displacement locally on these parts is visualized. There, we observe a high positive deviation (dark red) in the inner hood and on a local region on the firewall. This means

that the x-displacement, that is the intrusion, is much higher on the reference compared to the current variant.

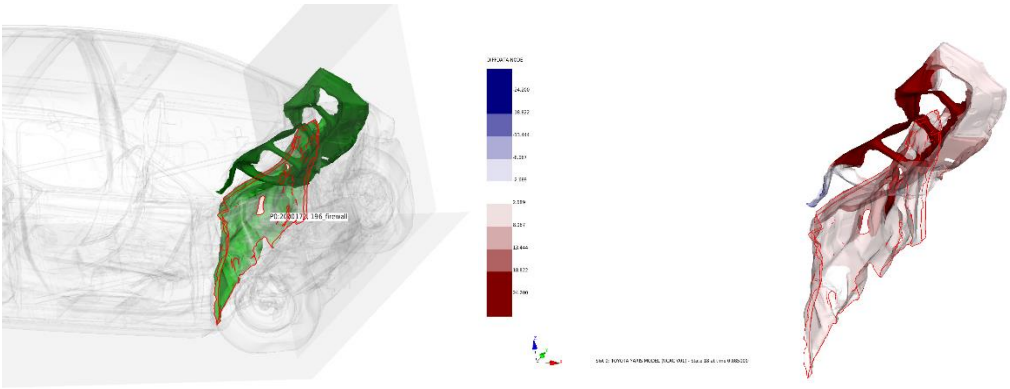

Figure 4: Most affected parts filtered out by SimCompare using threshold 0.227 and L1 norm (left), together with the distribution of the deviation in x-displacement on these parts (right).

Figure 5 shows the deviation in x-displacement again in detail for the part firewall (PID 2000172). There are high positive deviations in a local region (dark red) visible and also negative ones (dark blue). This means, the intrusion is much higher in the mid-upper part (dark red) of the firewall for the reference, while the current design has more intrusion at the borders of the firewall (dark blue).

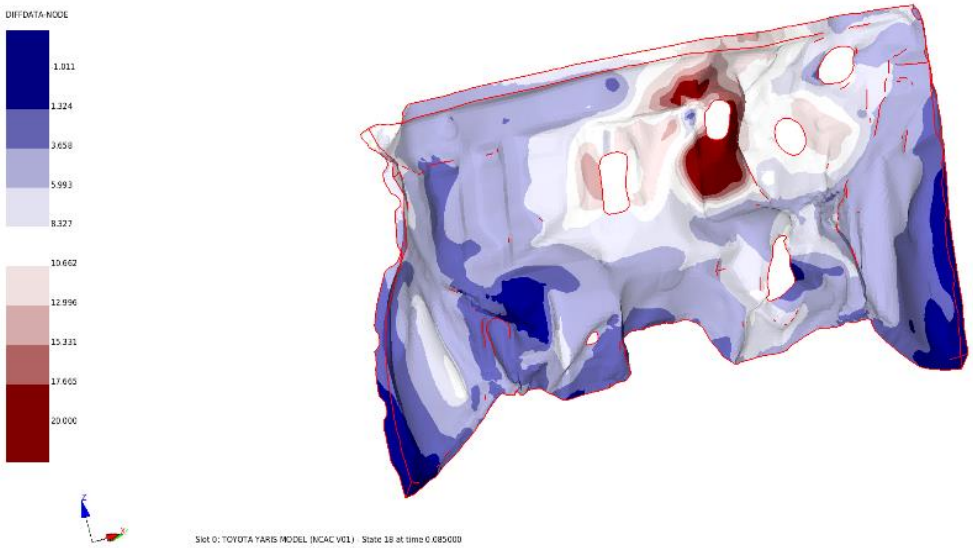

Figure 5: Deviation in X-displacement between reference and current variant computed by SimCompare and visualized by SimCompare GNS Animator plug-in. The intrusion is much higher in the mid-upper part (red) of the firewall for the reference, while the current design has more intrusion at the borders of the firewall (dark blue).

Additionally, the raw distribution of the x-displacements for the reference and the current variant on the firewall is given in Figure 6 for validation of the results.

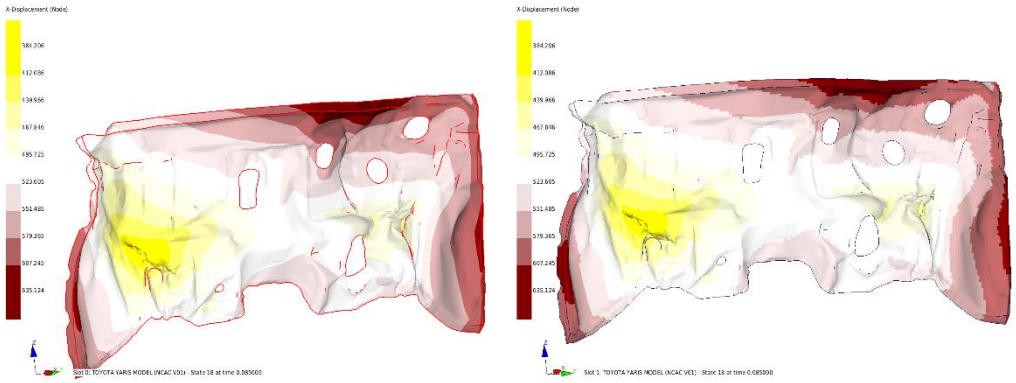

Figure 6: X-displacement on the firewall (PID 2000172) of reference (left) and current variant (right). A comparison of these two simulation results shows higher intrusion in the mid-upper part of the firewall for the reference.

In summary, we have seen that the proposed framework for automatic local event detection is able to identify the design measures applied and provides a possibility to filter the most affected parts automatically.

6. Future Work

In this work we focussed on one time step for simplicity, but an extension of the framework to analyse all timesteps simultaneously is straightforward. This enables not only the detection of local events, but also the time evolution of the event, in particular when a deviation on a part starts to appear. Based on the automatic design and event detection, we envision future work on machine learning approaches that assists with establishing cause and effect relations for detected design changes and local events.

7. Conclusion

We proposed a framework towards automatic local event detection in crashworthiness analysis results which provides a comparison of two FE simulation results based on arbitrary node or element data functions. The parts with the largest differences are automatically highlighted and stored in a structured data representation. We consider the presented framework as an important step towards automatic design and event detection in an overall simulation data analysis workflow. In particular, it allows to systematically analyse a design development tree to detect interesting deviations in the crash behaviour caused by design changes. The employed structured data representation highly simplifies the documentation for example in simulation

data management tools, as well as the further analysis of the observations found by for example artificial intelligence approaches. Additionally, it builds the basis for automatic pdf report generation, which allows the creation of human readable documentation of the changes and results.

8. References

Bohn, B., Garcke, J., Iza-Teran, R., Paprotny, A., Peherstorfer, B., Schepsmeier, U., Thole, C.-A. (2013). *Analysis of car crash simulation data with nonlinear machine learning methods:* International Conference on Computational Science ICCS, 621–630. 2013.

Diez, C., Kunze, P., Toewe, D., Wieser, C., Harzheim, L., & Schumacher, A. (2017). *Big-Data based rule-finding for analysis of crash simulations:* In World Congress on Structural and Multidisciplinary Optimization, 2017.

Garcke, J, Iza-Teran, R. (2017). *Machine learning approaches for data from car crashes and numerical car crash simulations*: In NAFEMS World Congress 2017.

Garcke, J., Pathare, M., Prabakaran, N. (2017). *ModelCompare*: In Griebel, M., Schüller, A., Schweitzer, M. (eds) Scientific Computing and Algorithms in Industrial Simulations. Springer, 199–205, 2017.

Iza Teran, R. (2014). *Enabling the analysis of finite element simulation bundles*: Internat. Jour. for Uncertainty Quantification, 4(2). 95-110, 2014.

Kracker, D., Garcke, J, Schumacher, A. (2020). *Automatic analysis of crash simulations with dimensionality reduction algorithms such as PCA and t-SNE:* In 16th International LS-DYNA Users Conference 2020.

Meywerk, M. (2007). *CAE-Methoden in der Fahrzeugtechnik*: Springer, 2007.

Steffes-lai, D, Iza-Teran, R., Klein, T. N., Garcke, J. (2021). *Cluster-based metamodeling:* In NAFEMS World Congress 2021, NWC21-379.