A Knowledge Based Engineering approach to automation of conceptual design option selection

E.J. Schut*, and M.J.L. van Tooren †
Delft University of Technology, Kluyverweg 1, 2629 HS Delft, The Netherlands

The Multidisciplinary Design and Optimization (MDO) process can be supported by partial automation of analysis and optimization steps. Design and Engineering Engines (DEEs) are a useful concept to structure this automation. Within a DEE a product is parametrically defined using Knowledge Based Engineering (KBE). This parametric product model requires an initial design solution before multidisciplinary optimization can be performed. The Initiator component of the DEE is responsible for finding this initial design solution. However, at the start of the design process the designer is faced with too many requirements, too many design properties, a too complex model, and too little resources. By using a simplified design problem the designer is able to find an initial design solution. Via iteratively adjusting the level of simplification the designer tries each time to find a feasible solution, in order to eventually find the initial design solution. This process highly iterative, requiring the same design knowledge to be accessed multiple times. To support accessibility of design knowledge surrogate models can be used that capture the optimized feasible solutions, representing the design space. At the same model aggregation level design options have identical objectives, based on which the optimal design option can be selected automatically. An implementation of the Initiator, featuring this methodology, is used in a sample DEE for aircraft vertical tail design. The selection process is implemented for panel structural design, based on four design options.

I. Introduction

In 2002, NASA and the Advisory Council for Aeronautics Research in Europe produced two relevant documents, the Aeronautic Blueprint and the Strategic Research Agenda, where the first century of aviation is briefly analyzed and the technological challenges for the next 20 years are set. They expect that in a couple of decades the aeronautic systems will differ from today’s systems at least as much as the actual systems differ from those of 1930. The aeronautic community will have to face such a challenge in a problematic socio-economic scenario, where the availability of economical and intellectual resources is shrinking upfront the increasing complexity of demanded products. Support can be found in the advances in computer technology, which create increasingly cheaper resources. However this shifts the balance between the computer and the human, making human knowledge relatively more valuable.

A fundamental paradigm shift is required to pass to a new knowledge based vision of business, where knowledge needs to be engineered and managed as a key business asset. Knowledge Management (KM) and Knowledge Based Engineering (KBE) represent two organizational and technical disciplines that can support this knowledge paradigm shift and help companies to retain their competitive advantage in the engineering market. Knowledge Based Engineering (KBE) is a technical implementation of Knowledge Management (KM) that enables knowledge reusability in repetitive tasks, such as in preparation and execution of analyses. A KBE application is a collection of knowledge rules that supports a human decision process. La Rocca points out that in building a KBE engineering application the capturing of knowledge rules can be performed best via following the normal engineering practices. The basis of knowledge-based systems is that knowledge is represented in an explicit form and used to reason about solving a problem and it provides the content and organization of design knowledge that can be used to generate a new product. KBE can be applied well in engineering design, featuring knowledge intensive products, like factories, ships, and aircraft.

* Researcher, faculty of Aerospace Engineering, Member AIAA
† Professor, faculty of Aerospace Engineering, MDO TC Member AIAA

American Institute of Aeronautics and Astronautics
Typically, engineering design (e.g. aerospace, mechanical engineering) involves the design of complex real world systems. In principle, engineering design aims at finding a physical relation between requirements, properties, and behavior that meets the requirements. Historically, to solve the engineering design problem, the design process encompasses multiple design disciplines (e.g. structures, aerodynamics), multiple design phases (e.g. conceptual, detail), and multiple levels of detail (e.g. macro, micro). The design cube, illustrated in Fig. 1, can visualize this three-dimensional design problem. The field that addresses such problems is the field of Multi-disciplinary Design and Optimization (MDO).

![Figure 1. The design cube (left) and the implications of problem decomposition on the design cube (centre and right)](image)

The complexity of the design problem faces designers with too many design options, too many tools, and too little time, to initially find a proper set of design options that describe a final feasible solution, and at the beginning of the design process little or no information is available on the product and at a higher level no mathematics are available that describe the problem. To attack these problems, a designer uses problem decomposition (see Fig.1) and problem simplification (see Fig.2). To mathematically solve the problem, problem decomposition is used to find a primitive level where the relationship between the requirements, properties, and behavior can be described physically. However, to be able to apply problem decomposition the designer must have a profound idea of what the solution could look like, requiring the designer to define all relevant model properties. Next to that, a new design solution implies a different property assembly and thus different behavior. However, in order to find a feasible design solution all product behavior must be known.

![Figure 2. The design initiation problem, simplification of the design problem](image)

In a previous paper by the authors on this topic, it was shown that KBE allows parametric modeling in the optimization sense, the concept of Design and Engineering Engines (DEE) was explained. The paper concluded that initiators must be developed for all aircraft elements of importance in the conceptual design stage. These initiators are responsible for the selection and feasilization of the design parameters of the parametric product model, providing an initial design solution based on which the global multidisciplinary optimization can be started. In another paper by the authors an implementation of a structural Initiator is discussed. The paper elaborates on the redesign of a vertical tail structure. In order to find an initial feasible solution, the concept of feasilization was introduced, which proved useful. From the design process followed that over eighty percent of the time was used to
resize the structure to the loads, which would have be considerably more if more design options would have been included. Next to that, many panel design problems were similar, but anyway calculated separately. The designer would be greatly helped if the accessibility if the design solution knowledge is increased.

To increase knowledge accessibility, surrogate models can be used. The model can store the required design knowledge directly, linking the requirements directly to the optimal feasible solution. The advantage is that during the design process no optimization processes are required to access the optimal feasible solution, note that the optimization processes are to be performed before the design process. An extra advantage is that the surrogate models based on optimization processes solutions can be created in a controlled environment.

This paper describes the development of a conceptual design methodology that aims at increasing the design knowledge accessibility. Via automation of the design option selection process a method is developed that uses surrogate models to capture design knowledge. First the concept of the Initiator is discussed in section III. The paper continues with a practical application of the methodology in conceptual design of a wing-type structure in section IV, and V. In section VI, results are presented, which is followed by the conclusions and recommendation in section VII.

To make this paper self sufficient first the DEE concept and the concept of feasilization are discussed shortly.

II. An overview of the DEE concept

A Design and Engineering Engine (DEE) is defined as an advanced design environment that supports and accelerates the design process of complex products through the automation of non-creative and repetitive design activities. Fig. 3 shows the concept of the DEE.

The main components of the DEE are:
- **Initiator**: Responsible for providing feasible starting values for the instantiation of the (parametric) product model.
- **Multi-Model Generator (MMG)**: Responsible for instantiation of the product model and extracting different views on the model in the form of report files to facilitate the expert tools.
- **Analysis (Expert) tools**: Responsible for evaluating one or several aspects of the design (e.g. structural response, aerodynamic performance or manufacturability).
- **Converger & Evaluator**: Responsible for checking convergence of the design solution and compliance of the product’s properties with the design requirements.

![Figure 3. The Design and Engineering Engine (DEE); left the main design process flow; right the Multi-model generator and the analysis tools.](attachment:image.png)
A detailed discussion of the DEE concept can be found in [2]. The definition of the product, i.e. the problem to be addressed by the DEE, is based on High Level Primitives\(^2\) (HLPs). These are functional building blocks, which allow the user of the DEE to define a product in a certain product family, which encompasses a structured set of HLPs. These functional blocks are basically sets of rules that use parameters to initiate objects that represent (part of) the product under consideration or to apply an engineering process to the initiated object. The object oriented approach allows the representation of the product and engineering process structure. Different classes of primitives are used to instantiate a certain model, which are in case of an aircraft; wing, fuselage, engines, and connection element primitives. The HLPs can be interpreted as building blocks, like rubberized LEGO\(^{TM}\), which can be individually tailored due to their fully parametric definition. The KBE environment gives access to a parametric geometric modeler. This allows the rule base to perform all geometric operations normally available in a CAD program. The DEE concept assists the designer in the search for a physical relation between top-level requirements, product properties, and behavior, see Fig. 4. Although behavior is also a product property, the term is used to differentiate between the design variable vector \((x)\), and design response vector \((y)\), related by: \(y = f(x)\). In this paper properties refer to \(x\) and behavior relate to \(y\).

![Figure 4. A DEE assists the designer is finding a physical relation between requirements, properties, and behavior](image)

The Initiator component of the DEE is responsible for providing an initial design vector that is (close to) a feasible solution of the multi-disciplinary design optimization problem. Although the result of the Initiator does not necessarily provide a feasible solution of the actual problem, it provides the best initial design vector that can be obtained with information and engineering knowledge available at the start of the global multidisciplinary optimization problem.

At the start of a design the designer has little to no knowledge of the design. Normally, the designer simplifies the design problem to find an initial feasible design solution (start vector). A simplified design problem is obtained via a reduced set of the requirements, simplification of the design properties (design options), and making use of lower fidelity models (schematic models) to describe the behavior. Iteratively, the designer decreases the level of problem simplification to eventually find a feasible solution that meets all the requirements.

Since no mathematics are available at a higher level, the designer uses problem decomposition, dividing the design problem in a set of primitive levels. Per primitive level, from lowest level primitive to highest level primitive, the design problem is simplified, and solved via trial and error methods (e.g. optimization methods). Trial and error methods are used to size the design variables and to select the optimum set of design parameters. Design variables specify difference within a solution topology (continuous variables), e.g. change in wing span or skin thickness. Design parameters specify differences in solution topology (discrete variables), e.g. on a high aggregation level one can be comparing a blended wing configuration with a conventional wing configuration. On a low aggregation level this may imply comparing riveting with bolting. Iteratively the primitive levels are addressed, from low level to high level primitive, such that a feasible solution is found at holds at all primitive levels.

This process of finding an initial feasible solution is called feasilization\(^3\). From the modeling of this multi-level primitive design process it came clear that the Initiator itself is a collection of DEEs, each with a different problem simplification. This is explained next.

### III. Design Initiator

#### A. Problem simplification process

The feasilization process captured by the Initiator component of any DEE can in fact be modeled by one or multiple DEEs, see Fig.5. Each DEE has a unique level of problem simplification. The number of DEEs depends on the top-level requirements, the problem complexity, and the available resources. During the design process these DEEs are applied sequentially. The order in which they are applied depends on the DEE result. If a certain DEE does not find a feasible solution space, the previous DEE process must be repeated, taking into account the infeasible design space. This process continues until the final DEE process yields a feasible solution.
The feasilization process clearly features an *iterative nature*. An useful approach that can be used to model this iterative nature, is called the *iterative approach*\(^4,5,6\), which is commonly applied in software engineering. This emphasizes the need of a “*damage tolerant*” design process. The implementation of the feasilization process should be such that it presumes that “unsuccessful” design iterations will occur.

**B. Problem decomposition process**

As stated before, the designer uses decomposition to find a physical relation between requirements, properties, and behavior. To solve the top-level problem, multiple primitive solutions can be selected with different topologies, illustrated in Fig.6. In many cases, each of these solution feature a comparable problem, since it is still not possible to physically describe the relation between requirements, properties, and behavior. These problems have again a set of different primitive solutions with different topologies. This process continues until the relations can be described physically. The number of primitive levels depends on the complexity of the solution.

![Figure 5. The Initiator component of a DEE consists of multiple DEEs of simplified problems](image)

The relation between the primitive levels is such that the selected primitive solution (e.g. design option) specifies the problem domain of the lower level primitive selection. If the primitives are designed separately, the variables (both continuous and discrete) of a certain primitive design problem are constants in the lower level primitive design problem. The multi-level primitive model is illustrated in Fig.7. The methodology used by the human designer to select a certain design option is addressed in section IV.

![Figure 6. Relation between problem domain and solution domain](image)

![Figure 7. Relation of the design space and solution space for multiple primitive levels](image)
C. Generic primitive object

From the implementation came clear that each primitive could be modeled with the same basic object. The primitive object is build upon this basic object. The basic object has parent and children, and six basic attributes; properties, design space, test cases, behavior, criteria, and child relations. The properties specify the state of the object. The design space defines which properties can be changed, and within which bounds. The test cases specify the objects expected operation requirements. The behavior enables the engineer to relate properties and test cases. The criteria specify the behavioral requirements. The child relations define how top level test cases are cascaded to each of the children, based on their properties. This can be represented in formula form by:

\[
\text{criteria} (\text{behavior} (\text{properties} (\text{designSpace}), \text{testCases}))) = \text{designValue}
\]  

(1)

The analysis of an object to get the design value consists of several steps. First the new property values specified by the design space interrogator (e.g. optimization routine) are set. These properties are then cascaded down and then up to all the primitives. Then the test cases are cascaded down using the child relations. Now the behavior of the primitives can be calculated using the methods. The result is then tested against the criteria to obtain the design value. Finally, the object evaluator (e.g. optimization routine) decides if a new design will be investigated. The process is illustrated in Fig.8.

![Figure 8. Information flow in the conceptual design process](image1)

IV. Design options

A. Design approach

Commonly products are separated into two categories, routine and non-routine designs. Routine design encompasses all designs that are recognized as not being different from previous designs in their class in any substantive way and consequently non-routine designs are the designs that do differ substantially. Gero and Maher\cite{5} introduce a sub-categorization of non-routine designs in innovative and creative designs. They recognize an innovative design as a design where substantial difference is caused by design option variable values that are outside the commonly used range and creative designs as designs where the substantial difference is caused by the introduction of new design options, illustrated in Fig.9.

![Figure 9. Different design categories](image2)
The DEE supports the human designer in finding new design solutions. A human designer uses four methods to find a new design solution: combination, mutation, analogy, and first principles. Combination involves importing parts from various designs and combining them into a new design. Mutation involves a modification of a property. Design by analogy involves making associations to generalizations outside the current domain, and design by first principles, based on the laws of nature.

These four methods can be divided into two general design approaches. The first three are all based on making new associations between existing design solutions, thus based on existing design knowledge. These methods can be applied to search the existing design knowledge to find new solutions. However, if new design knowledge is required, designing from first principles provides the engineer with a means to investigate the requirements divorced from previous design decisions. Only design by first principles can be used to describe physically a new relationship between the requirements, properties, and behavior.

Normally, in a design process both approaches are applied to find a new set of design options that describe a feasible design solution. A drawback of design by first principles is that it requires a solution validation at encompasses all primitive levels, where the other methods can be used to use a known relationship between the requirements, properties, and behavior in another solution at any primitive level. The designer has to validate only higher level primitives, not lower level primitives, since they were already validated in a previous design. Consequently, the second approach appears to require fewer resources, but is constrained by the existing design knowledge.

B. Selection of the design options

Commonly, since the designer is faced with too many design options and has too little resources, the designer uses a selected set of all design options to be investigated, the size of the set depending on the problem complexity and the available resources. These design options are sized to find the optimal solution. Evaluation of the solutions will reveal which design option performs best, as defined by the objective function. If no design option yields a feasible solution, a new selection must be made.

Consider a design problem having two design options. The design problem is constrained by a set of parameters that can vary per problem, but not within one problem, that span the design domain. This is illustrated in Fig.10 for two parameters: X, and Y. Sizing a finite number of experiments, a combination of parameter values, that fully map the design domain, can be used to interpolate and extrapolate (to a certain degree) any other combination of parameter values. In this fashion one surface can be obtained per design option, relating the parameter values to the objective values. This can be done for any number of design options, which can all be represented in the same design domain. The result is a surface that is build up from multiple design option surfaces, directly linking the parameter values to the best objective, and corresponding design options.

For instance, when designing a panel structure, the length and width of the panel are fixed. However multiple combinations of length and width are possible. When from experience follows that panel dimensions vary within a fixed domain, the designer can select a finite number of experiments that map the domain. Sizing of each of these panels delivers the optimal combination variable values, and the optimum objective value. These results can be interpolated to obtain an optimal solution surface. The combination of multiple surfaces results in a multi design option surface, based on which the optimal design option can be selected automatically.

Figure 10. Integrating design options in a single design domain
C. Capturing design knowledge

To enable the designer to directly access the design knowledge, a relation between the requirements, properties, and behavior must be captured. Surrogate models are widely used for this purpose. Instead of first principle based behavioral models, an abstraction is made of this model that directly links requirements to properties and behavior. The evolution of evaluation models is illustrated in Fig. 11.

![Figure 11. Evolution of evaluation models](image)

Surrogate models can be used to physically describe the relations for higher-level primitives. Via design by first principles the design knowledge for every lowest level primitive can now be captured. This means that at one level higher design by combination can be used to find the optimal solution. Using surrogate models also, this design solution knowledge can be captured. Making design by combination at one primitive level higher possible. This process can be continued until the relation between requirements, properties, and behavior is captured at the highest level primitive, delivering a multi primitive level surrogate model (see Fig. 12). Note that the human designer it still responsible for the selection of the design options that will be included in the model. The benefit is that, in principle, the problem could be analyzed on a higher abstraction level, saving calculation time.

![Figure 12. Multi primitive level surrogate model; captures design knowledge at all primitive levels of the design, enabling the designer to interrogate the design space at multiple primitive levels.](image)

Another advantage is that new design options do not require the development of a complete new model. At the level that the design options are added, a surrogate model must be again trained to capture the new design knowledge. Only the higher level primitives that embody the lower level primitive then be updated to compensate for the extended design space, illustrated in Fig.12.

The implementation of this methodology in the Initiator component is now discussed for wing-type structures. In this paper the surrogate models are not included in the implementation. Future developments will focus on the integration of the surrogate models.
V. The wing-type structure Initiator

The methodology introduced in III and VI is applied in the conceptual structural design of wing-type structures. In a previous paper\(^7\) by the authors, an elaborate discussion and implementation of the feasilization process was presented, based on a single design option. Cerulli presented in [8] a preliminary version of the structural design tool, which was used within a framework for vertical redesign of tools encompassing a dynamic load analysis tool, KBE modeling tool, and a Finite Element (FE) modeling tool.

A. Problem simplification

The simplified design problem is obtained via a reduced set of the requirements, simplified design options, and simplified behavior. The requirements are reduced to encompass flight maneuver, and deformation loads, in this case as a result from a yaw motion, a minimum weight requirement, and design criteria that encompass only strength and stability criteria.

Since only maneuver loads are considered, the design options are simplified to encompass only the central box, for which these are critical. The feasilization of the trailing and leading edge would require another requirement set, e.g. including load cases like bird impact.

Included structural behaviors are rib crushing, panel buckling, skin buckling, stiffener buckling, and material elasticity.

B. Problem decomposition

For problem decomposition it is useful to partition the problem according to the requirements, in case of simplified structural design based on the design parameters and expected load paths.

Commonly, the loads on the wing and empennage structures increase from tip to root, the structure is decomposed in a set of trunks that vary in sweep and aspect ratio. The trunks consist of a set of constituent section primitives, which specify the local cross-section (wing profile), based on the rib positions. The sections contain a set of box primitives, representing the leading edge, centre, and trailing edge. NuU\(^{10}\) states that the maneuver loads are first taken up by the skin panels. Via the ribs the loads are finally transported to the spars. Within the simplified problem, skin, rib and spar panels all have an identical function, namely transportation of loads. Hence, decomposition of a HLP wing-type structure delivers a set of panel primitives. This decomposition is illustrated in Fig.13.

![Figure 13. Problem decomposition: Transformation of a high-level wing-type model to multiple low-level panel models](image)

The decomposition is one of many. This decomposition is specifically tailored to the structural design optimization problem, in case of other design problems the decomposition could be different. In this paper the implementation of the methodology is implemented from material primitive to panel primitive. In Fig.14 the relation between panel design and wing-type design is illustrated. Further development will focus on an implementation of the methodology for the complete wing structure.

![Figure 14. Decomposition of a wing structure delivers a set of panel structures](image)
VI. Panel structure Initiator

A. Problem simplification

The design criteria are requirements on strength and stability. The design options are simplified to a flat rectangular panel, designed to a set of test cases of two-dimensional load cases. In order to calculate the strength and stability behavior, the engineer requires geometry, and stiffness properties. The latter can be material properties or more general, the ABD matrix, which relates structural loads to structural deformation. In the implementation the ABD matrix is used.

The panel design requirements are reduced to a minimum weight requirement and a set of two-dimensional load cases that it must be able to sustain, that specify the minimum required allowable loads of the structure. A load case can be any combination of tension or compression with shear. In this paper only one axis is considered to be loaded, biaxial loading conditions are out of scope. Strength and stability criteria are included, but no stiffness constraints and design life constraints related cyclic loading are taken into account.

B. Problem decomposition

The panels can be further decomposed, based on the chosen structural concept. It is either stiffened or not stiffened, separated based on their structural function. The skin and stiffener consist of one or multiple plate primitives. In case of a blade stiffened panel, both skin and stiffener contain a single plate primitive. These plate primitives contain one or multiple layer primitives, varying in thickness. In case of a sandwich panel with isotropic facings, both core and face are represented by one layer primitive. The layers consist of one or multiple ply primitives with identical thickness, but different fiber orientations. A ply consists of a single material primitive that specifies material properties. This decomposition is illustrated in Fig.15.

From Fig.8 follows that property information flow from the children to the parents, and the test case information from parent to the children. The advantage of using the ABD matrix is that the parent properties are obtained via a relatively simple addition of the children ABD properties, based on the child relations. Also the test cases are determined relatively easily since the ABD matrix directly relates loads to deformation, and vice versa. In this manner the parent load cases can be translated to parent deformations, giving the child deformations. These deformations can then be transformed again to load cases.

Figure 15. Problem decomposition; Transformation of a high-level panel model to multiple low-level panel materials

Figure 16. Panel design options: flat, sandwich, blade-stiffened, and z-stiffened.
The panel design options are simplified by an approximation of the real panels by flat, rectangular panels is done most conservatively by taking the smallest dimensions in width and the largest dimension in the length. The panel design options are specified by structure type, and material type. In this paper a set structure type design options are addressed, namely; un-stiffened, blade-stiffened, z-stiffened, and sandwich, illustrated in Fig.16. Stacking sequence and material are held constant. The layer stacking is fixed at [+45,-45,0], and in each case the same orthotropic material is selected. The sandwich panel is symmetric and has symmetric face sheets, and in the current implementation it is based on an isotropic core, and face dimpling and shear crimpling are not evaluated. The design parameters are panel length, and width. The design test cases are a set of tension or compression, and shear loads. The design constants are length, width, normal load, and shear load, and the design variables are layer thicknesses, stiffener spacing (if relevant), stiffener height (if relevant), and stiffener flange width (if relevant).

C. Behavior modeling

The panel behavior is simplified by taking into account only structural local buckling, global buckling, shear buckling, face wrinkling, and material failure strengths, see Fig.17. No bending coupling is included in the calculation of the buckling loads. An elaborate discussion including finite element models and verification results on the schematic models that describe a blade-stiffened panel can be found in [3]. In this paper model verification is out of scope.

\[ \sigma_{\text{flexural panel buckling}} = \frac{\pi^2 D_{11}}{\text{length}^2} \]  
(3)

\[ \sigma_{\text{local skin buckling}} = \frac{2\pi^2 \text{thickness}}{\text{width}^2 \sqrt{D_{11}D_{22} + D_{33}}} \]  
(4)

According to Rothwell\(^9\), in shear, only the skin will buckle, so no shear buckling constraints are defined for the stiffener. The shear buckling load of a long plate subjected to pure shear is estimated using an ESDU\(^{11}\) paper. The plots on long plates subjected to a shear load are parameterized and included via an equation, which describes the relation.

The total buckling load depends on the load condition\(^9\). In pure compression or shear the above formulae are used. In case of compression and shear, and in case of tension and shear, the skin web buckling stress \(\sigma\) is calculated by respectively:

\[ \sigma_{\text{local skin buckling}} = \sigma_{\text{compression}} + \sigma_{\text{shear}}^2 \]  
(5)

\[ \sigma_{\text{local skin buckling}} = 0.5 \cdot \sigma_{\text{compression}} + \sigma_{\text{shear}} \]  
(6)

Figure 17. Panel behavior considered in the design
3. Plate

Local stability of a rectangular web with three edges simply supported and one edge free is evaluated with formula 4. In case of compression local stiffener web buckling is evaluated with a formula based on a formula for isotropic materials for the initial buckling load of a long flange that is adapted for composite materials:

\[
\sigma_{\text{local stiffener buckling}} = \frac{2K}{I_0} = \frac{2 \cdot (4 \cdot \text{width} \cdot D_{31})}{\frac{1}{2} \cdot \text{width}^3 \cdot \text{thickness}}
\]  

(7)

Here, K is the torsional stiffness, and \(I_0\) is the polar moment of inertia about the root. In case of isotropic materials this formula reduces to the well-known formula:

\[
\sigma_{\text{local stiffener buckling}} = \frac{1}{2(1+\nu)} E \left( \frac{\text{width}}{\text{thickness}} \right)^2 \approx 0.385 E \left( \frac{\text{width}}{\text{thickness}} \right)^2
\]  

(8)

4. Plate

Two formulae suggested by Kollár \cite{12} are used to calculate the wrinkling stress. In the design the most critical is included in the evaluation, respectively for short wave and long wave wrinkling of a sandwich with an isotropic core and composite facesheets, the following formulas are used to calculate the initial wrinkling stress \(\sigma\):

\[
\sigma_{\text{face}}^\xi = 1.5 \cdot \sqrt{\frac{2 \Psi_{\xi} \sigma^2}{\pi^2}}, \text{with : } \sigma = \frac{2 \pi E_c}{(3-\nu_c)(1+\nu_c)}
\]  

(9)

\[
\sigma_{\text{face}}^\xi = 2 \cdot \sqrt{\frac{E_c}{\Psi_{\xi} c/2}}
\]  

(10)

Here, \(c\) is the core thickness, and \(\Psi_{\xi}\) is the bending stiffness of the facesheet in \(\xi\) direction. It depends on the load resultant, which has angle, \(\alpha\), with the x-axis, and is defined as:

\[
\Psi_{\xi} = D_{11}^\text{face} \cos^4 \alpha + D_{22}^\text{face} \sin^4 \alpha + (2D_{12}^\text{face} + 4D_{66}^\text{face}) \cos^2 \alpha \sin^2 \alpha + 4 \cos^3 \alpha \sin \alpha D_{16}^\text{face} + 4 \sin^3 \alpha \cos \alpha D_{26}^\text{face}
\]  

(11)

5. Ply

The stresses in the layer plies are determined with the Classical Laminate Theory using the A matrix, based on the ply stiffness matrix S, and angle between its principal material axes and the loading axes system. Ply failure is evaluated with the Tsai-Hill criterion:

\[
TH = \frac{\sigma_x^2}{S_X} + \frac{\sigma_y^2}{S_Y} + \frac{\tau_{xy}^2}{S_{XY}}
\]  

(2)

With \(s_X, s_Y,\) and \(s_{XY}\) as respectively the maximum allowed normal stress in x and y direction, and maximum allowed shear stress.

D. The design problem

The panel model is defined by a set of design variables, \(\bar{x}\), and a set of fixed design constants, e.g. panel length and width, material, structural concept, load intensities, and layer stacking sequence. The optimization can be formally described as a combination of an objective function, a set of equality, \(c_{eq}\), and inequality constraints, \(c\), and bounds, \(\bar{b}, \bar{u}\), on the design variables:

\[
\min_x f(x) \quad \text{subject to : } \quad c_{eq}(\bar{x}) = 0 \quad c(\bar{x}) \leq 0 \quad \bar{b} < x < \bar{u}
\]  

(12)
The objective function $f$ evaluates the weight of the panel; the weight calculation depends on the design option. The flat design option has 2 Design Variables (DVs), the sandwich design option has 3 DVs, the blade-stiffened design option has 6 DVs, and the z-stiffened design option has 8 DVs. Inequality constraint functions are used to evaluate the material strength and the panel stability. Strength, evaluated with Tsai-Hill (TH), and stability, evaluated with the initial buckling stress $\sigma_{\text{buckling}}$ and the applied stress $\sigma_{\text{applied}}$ are transformed to a set of inequality constraints, given by formulae 13 and 14.

\[ c_{\text{strength}} : \text{TH} - 1 \leq 0 \]  
\[ c_{\text{stability}} : \frac{\sigma_{\text{applied}}}{\sigma_{\text{buckling}}} - 1 \leq 0 \]  

VII. Implementation and results

A. Design problem implementation

A preliminary version of this concept was programmed in Matlab. Although Matlab supports both fast calculation and had an optimizer available that suited the problem, file operations were stalling further development. It became increasingly difficult to extend the existing code. To support development, building extensions and performing both calculations and file operations, the concept is implemented using the Object Oriented (OO) Python programming language.

As an optimizer a python package called COBYLA is used, a derivative free constrained optimization package that uses linear approximation to find the objective, part of the Scipy optimization package. The algorithm requires each constraint to be specified in a single function. However, in the problem the constraints values are a result from a single model evaluation, and can not be evaluated separately. A constraint evaluation would result in a complete model evaluation. This increases the number of model evaluations and thus calculation time. To safe resources the problem is rewritten to a single objective evaluation:

\[
\min_{x} \ obj = f(\bar{x}) + \sum_{i=0}^{n} (c_{i}(\bar{x}) + 1)^{2} + \sum_{y \in \mathbb{Y}} M \cdot (c_{y}(\bar{x}) + 1)^{2}
\]

subject to: $\underline{b} < \bar{x} < \overline{b}$

Here $M$ is a large number. Constraint violation and non-optimality conditions are included as penalty functions within the objective function. Although this also deteriorates the optimization process the required calculation time is reduced. COBYLA is used since it was readily available and relatively easy to integrate.

B. Design of Experiments

To fully map the design space, a relatively simple Design of Experiments (DoE) was programmed to fit a number of experiments equally within an n-dimensional domain. The domain space of the panel, described in section V has four dimensions, panel length, panel width, normal load intensity (which can be both compression and tension), and shear load intensity.

Each design space dimension is represented by six experiments, with four dimensions this results in $6^4 = 1296$ equally spaced experiments. Each experiment is an optimization problem that is solved by COBYLA, using the in Python implemented concept. In total 5184 experiments were performed.

C. Results

A selection of the experiments is presented in Fig.18. The design options are compared for all combinations between of a set of constants, with the other two held constant. The graphs show to the designer all design options in a single design space. In this case the selection process could relatively simple be automated by selection of the concept having the lowest weight. However, it can happen that the designer is interested also in, for instance stiffness. Then the two design objectives could be plotted for a single design constant, to give the designer an idea of the design space. However, probably the solution is implicitly specified by a single higher-primitive level objective that specifies the optimal combination of lower-primitive level objectives.
During the design process a surrogate model could be used to map the optimal design solutions. The accuracy of the answer of the surrogate model should be within the accuracy of the model. As surrogate model neural networks or response surfaces can be used. Full automation of the design option selection process is not yet implemented.

The results show that within the used domain the solutions found by the optimizer are (almost) not influenced by changing the normal load intensity in comparison to the shear load intensity, which is probably the result of the selected domain. However, in all the figures the surfaces feature also unexpected changes, caused by a failing optimizer.

VIII. Conclusions and recommendations

The Initiator component, part of the Design and Engineering Engine (DEE) concept, has been elaborated and implemented for panel structures. Using design feasilization implementation of this component for the initiation of structural parts of an aircraft has been shown feasible taking into account multiple design options.

The concept of design feasilization solves a simplified design problem in order to initiate a higher fidelity problem. The feasilization design process encompasses design option selection, sizing, and evaluation. Problem simplification is achieved by selecting a subset of the requirements, simplification of the design options, and making use of approximate so-called schematic models for the sizing of the simpler solution, key elements used by the human designer to handle complex problems.
The examples show that the design option selection function of the structural engineer in principle can be automated using optimization techniques, surrogate models, and simple algebra. However, the obtained results can not be trusted, since the optimizer experienced difficulties in finding a solution, let alone the optimal solution. On the other hand, the visualization of the design option in a single design domain is possible and could be used by the designer to get a better understanding of the requirements and design options.

Shown was that by performing a set of experiments the design space can be mapped. The size of this set directly influences the quality of the mapping of the design space by any surrogate model. It is important that the solution estimate has the same contingency as its problem. The use of surrogate models is not absolute, but is a trade-off between number of required experiments for solution quality, and the number of performed experiments in the design process, the design time constraints, and the available calculation resources.

Future developments will aim at a full implementation and automation of the feasilization methodology in the Design and Engineering Engine (DEE) concept, including the design option selection and evaluation functions of the designer. The first development step will aim at integrating another optimizer into the Python environment.

References
1ACARE, (Advisory Council for Aeronautic Research in Europe), Strategic Research Agenda, Vol. 1+2 and executive summary, 2002
8Rothwell, A., “Structural Design and Optimisation II”, Faculty of Aeronautical Engineering, Delft University of Technology, TU Delft, 1999