



# Realization of a Framework for Simulation-Based Large-Scale Shape Optimization Using Vertex Morphing

Aditya Ghantasala<sup>1</sup>  · Reza Najian Asl<sup>1</sup> · Armin Geiser<sup>1</sup> · Andrew Brodie<sup>2</sup> · Efthymios Papoutsis<sup>2</sup> · Kai-Uwe Bletzinger<sup>1</sup>

Received: 17 July 2020 / Accepted: 29 January 2021 / Published online: 11 March 2021  
© The Author(s) 2021

## Abstract

There is a significant tendency in the industry for automation of the engineering design process. This requires the capability of analyzing an existing design and proposing or ideally generating an optimal design using numerical optimization. In this context, efficient and robust realization of such a framework for numerical shape optimization is of prime importance. Another requirement of such a framework is modularity, such that the shape optimization can involve different physics. This requires that different physics solvers should be handled in black-box nature. The current contribution discusses the conceptualization and applications of a general framework for numerical shape optimization using the vertex morphing parametrization technique. We deal with both 2D and 3D shape optimization problems, of which 3D problems usually tend to be expensive and are candidates for special attention in terms of efficient and high-performance computing. The paper demonstrates the different aspects of the framework, together with the challenges in realizing them. Several numerical exam-

---

Communicated by Zenon Mróz.

---

✉ Aditya Ghantasala  
aditya.ghantasala@tum.de

Reza Najian Asl  
reza.najian-asl@tum.de

Armin Geiser  
armin.geiser@tum.de

Andrew Brodie  
andrew.brodie@bmw.de

Efthymios Papoutsis  
efthymios.papoutsis@bmw.de

Kai-Uwe Bletzinger  
kub@tum.de

<sup>1</sup> Technische Universität München, Arcisstrasse 21, 80333 Munich, Germany

<sup>2</sup> BMW Group, Munich, Germany

ples involving different physics and constraints are presented to show the flexibility and extendability of the framework.

**Keywords** Shape optimization · Vertex morphing · Geometric constraints · Multi-physics optimization · Additive manufacturing

## 1 Introduction

Design optimization has been an indispensable part of the engineering design process across many disciplines; prominent fields include aerospace and airplane design. In this context, topology optimization has been used extensively to generate optimal designs under given conditions [2,5,39]. With the developments in recent times, shape optimization has also gained importance in simulation driven design and optimizations. [4,15,27,29] present some interesting applications of shape optimization. Traditionally in shape optimization, a parametrization of the shape of the object and corresponding parameters are used in optimization as design variables. For complex geometries, this parametrization procedure is tedious and can limit the freedom in shape optimization. In contrast to the above node-based shape optimization provides a higher degree of freedom, by taking the nodal coordinates of the discretized geometry as design variables.

The advantage of using the nodal coordinates as design parameters is twofold; first, it provides the largest possible design space for optimization, and second, it eliminates the necessity of using an explicit parametrization of the geometry. Previous works together with the original paper of [8] and the works [8,35] introducing vertex morphing regularization in the context of node-based shape optimization have shown promising results in different applications. Today, thanks to the improvements in the computational power and hardware, the number of design variables used in shape optimization have increased by many folds. This increase, along with an expanding number of application areas, is pushing the performance limits of the optimization techniques and the frameworks providing optimization and related capabilities.

In the wake of increasing usage and importance of node-based parametrization, an efficient and flexible framework offering respective functionalities becomes the need of the hour. The two essential characteristics of such a framework are its ability to include new algorithms and techniques, and the ability to extend the existing ones. Apart from these, it is vital that the framework is able to work with external solvers which provide the primal solution and sensitivity information required for shape optimization.

In this work, we present a variety of multi-disciplinary applications of shape optimization achieved with the framework along with a description of its important capabilities. The above-listed requirements of the framework are realized using implementations in C++, which then are exposed to Python. The implementations in C++ deal with computationally expensive tasks, whereas the python routines steer the optimization process. The python scripting layer gives the necessary flexibility and ability to prototype different optimization problems and algorithms rapidly.

Previous works on similar tools for a general optimization like pyOpt [31], GEMS [17], DAKOTA [14] though have several optimization algorithms readily available,

do not provide functionalities necessary for computing geometrical constraints, which arise in shape optimization problems. Apart from this, the lack of possibility to deal with different external solvers and parametrizations in a unified way makes them less suitable for shape optimization. Recently developed tools put forth like ABCD tool box [32] come close to the current work, but are developed with focus on specific applications and especially lack capabilities to handle node-based shape optimization. Though OpenMDAO [21] frameworks' objectives are parallel with the current work, it relies on applications and tools with extensive access to inner data structures to compute necessary quantities for optimization. modeFRONTIER from ESTECO is a commercial tool with a user-friendly GUI platform which has the similar capabilities as the current work. Ebes + jMetal a tool presented in [38] though can do multi-objective optimization, lacks capability to customize and specialize the provided objectives or constraints and depends heavily on external tools for this. On the other hand, tools for solving coupled multi-physics problems like EMPIRE [36] and preCICE [10] contain the ability to work with different solvers. However, their focus is on a different set of problems and are less suitable for advanced and sophisticated shape optimization problems.

The current framework used in the current work differs from the tools mentioned above in terms of its focus on requirements for shape optimization using vertex morphing parametrization. It also provides a flexible and generic interface to compute and include geometrical and other special types of constraints. It also uses a flexible methodology for the exchange of data required for optimization. These features also allow the usage of the current framework together with other tools, thus enabling a multi-physics optimization; more details are presented in Sect. 4.3.1.

The remainder of the paper first presents the theory of shape optimization and vertex morphing technology used in the optimization. The discussion of software concept and the details framework follows the theory. The paper concludes with the discussion of different application cases and results obtained using the framework presented, thus demonstrating its capabilities.

## 2 Shape Optimization Problem

The shape optimization problem following the node-based approach described in [26] may be defined as

$$\begin{aligned}
 & \min_{s_\Gamma} \quad J(X_\Gamma(s), \mathbf{Q}(X_\Gamma)) \\
 & \text{subject to} \quad \mathbf{A}_{i,j} s_j - X_{\Gamma,i} = \mathbf{0}, \quad i = 1, \dots, m_\Gamma, j = 1, \dots, m_s. \\
 & \quad \quad \quad g_j(X_\Gamma(s), \mathbf{Q}(X_\Gamma)) \leq 0, \quad j = 1, \dots, m_g. \\
 & \quad \quad \quad g_{\Gamma,j}(X_{\Gamma_g}) \in \mathbf{g}, \quad j = 1, \dots, m_{\Gamma_g}. \\
 & \quad \quad \quad h_j(X_\Gamma(s), \mathbf{Q}(X_\Gamma)) = 0, \quad j = 1, \dots, m_h. \\
 & \quad \quad \quad h_{\Gamma,j}(X_{\Gamma_h}) \in \mathbf{h}, \quad j = 1, \dots, m_{\Gamma_h}. \\
 & \quad \quad \quad \mathbf{r}_j(X_\Gamma(s), \mathbf{Q}(X_\Gamma)) = \mathbf{0}, \quad j = 1, \dots, m_\Omega + m_\Gamma.
 \end{aligned} \tag{1}$$

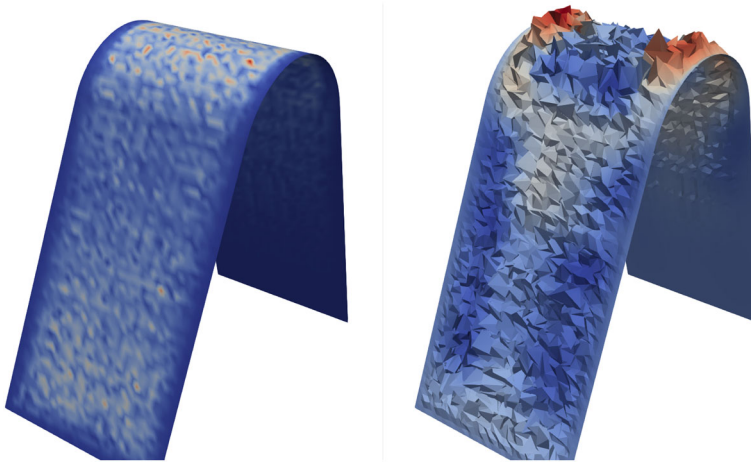
where  $\Gamma \subset \partial\Omega$  is the design boundary of the computational domain  $\Omega$ ; unless noted otherwise,  $\mathbf{X}_\Gamma$ ,  $\mathbf{X} \in \mathbb{R}^3$  denotes the vector of nodal coordinates of size  $3m_\Gamma$ ;  $\mathbf{A}$  is a generic mapping matrix which performs transformation between the design geometry  $\mathbf{X}_\Gamma$  and the design variables  $s$ ;  $J$ ,  $\mathbf{g}$  and  $\mathbf{h}$ , respectively, denote the objective function to be minimized, the vector of inequality constraints and the vector of equality constraints.

$\Gamma_g$  and  $\Gamma_h$  are the subsets of the boundary  $\Gamma$  which are subject to geometric inequality constraints  $\mathbf{g}_\Gamma$  and equality constraints  $\mathbf{h}_\Gamma$ , respectively.  $\mathbf{r}$  denotes the residual of the state/primal governing equations which may be nonlinear and  $\mathbf{Q}$  denotes the state/primal vector.

The geometric constraints can be enforced on the optimization problem in multiple ways. A direct approach is to formulate them point-wise, which means for each mesh point  $\mathbf{X}_{\Gamma,j}$ , scalar-valued constraints  $h_j$  and  $g_j$  are evaluated and satisfied individually. In context of node-based shape optimization with vertex morphing, this has been investigated by [4] for no penetration constraints. As an alternative, multiple geometric constraints of the same type can also be aggregated into one single constraint and therefore be treated in a weak sense. In such cases, they will not be fulfilled on every point, but in an average sense. This approach is used in the example presented in Sect. 4.2.1. A third option is to directly include the geometric constraint into the design parametrization, and a priori exclude infeasible designs [34]. A similar approach is used in the example from Sect. 4.1. In addition to these geometric constraints, the optimization problem can be subjected to any number of physics-based equality and inequality constraints. Literature provides robust methods for enforcing these constraints [7,11,16].

## 2.1 Vertex Morphing Technique

Gradient-based first-order optimization methods can be used to solve the node-based shape optimization problem presented above. Apart from being one of the fastest methods, they are also robust with minimal effort. Adapting these methods and considering that for the node-based approach, the solution of optimization problem requires the gradients, also called sensitivities, of the objectives and constraints with respect to the design variables  $s$ . Since the nodal coordinates are considered as the design variables, the sensitivity with respect to the nodal coordinates at each node on the design boundary  $\Gamma$  is required. For many classical objectives considered in engineering applications, these nodal sensitivities tend to be noisy and an optimization with noisy sensitivities produce an undesirable shape of the geometry, though it is optimal. Figure 1 shows an example noisy adjoint sensitivities calculated for strain energy and the corresponding optimal shape generated. To deal with this problem, in the current framework for optimization we use the vertex morphing parametrization technique proposed in [8] to consistently transfer the shape optimization problem from the spatial design space  $\mathbf{X}_\Gamma$  (the geometry space) to a new design space  $s_\Gamma$ , called control space, using a mapping operation. This will also filter the sensitivity field, thus smoothing it. Unless otherwise noted, in the following, the control field lives on the same discretization as the surface of the geometry. This allows the largest design space possible with minimal



**Fig. 1** Unfiltered noisy sensitivities and resulting optimal shape

to negligible modeling effort.

$$s_{\Gamma} = [s_{\Gamma,1}, s_{\Gamma,2}, \dots, s_{\Gamma,m_{\Gamma}}] \quad (2)$$

As defined in Eq. 1, the association of the discretized geometry with the discretized design space is expressed via the operator matrix  $\mathbf{A}$ . This operator maps  $s_{\Gamma}$  onto  $\mathbf{X}_{\Gamma}$  as follows:

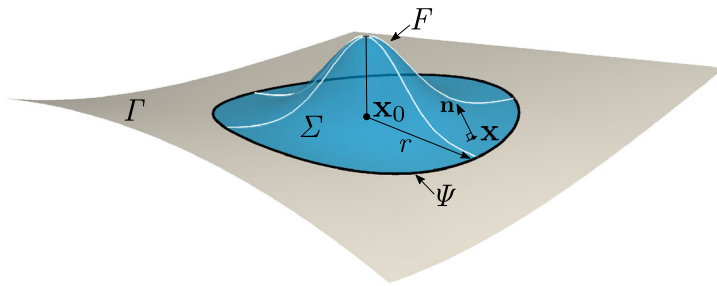
$$\mathbf{X}_{\Gamma} = \mathbf{A} s_{\Gamma} \quad (3)$$

In a continuous space, the three-dimensional geometry at point  $\mathbf{X}_0 = (X_0^1, X_0^2, X_0^3)$  of the optimization surface  $\Gamma$  is generated from the surface control field  $\mathbf{s} = (s^1, s^2, s^3)$  via a smoothing filter operation:

$$\mathbf{X}_0 = \int_{\Sigma(s,r)} F(\mathbf{X}, \mathbf{X}_0) \mathbf{s} \otimes \mathbf{n} \cdot d\mathbf{\Sigma} = \int_{\Sigma(s,r)} F(\mathbf{X}, \mathbf{X}_0) \mathbf{s} d\Sigma \quad (4)$$

where  $F$  could be any reasonable filter (kernel) function.  $\Sigma$  is the portion of  $\Gamma$  which lies within a sphere of radius  $r$  centered at  $\mathbf{X}_0$ , where  $r$  is the filter radius (assumed to be constant);  $\|\mathbf{X} - \mathbf{X}_0\|$  is the Euclidean distance to the center of the filter  $\mathbf{X}_0$ ;  $\mathbf{n}$  is the unit normal vector to the surface;  $d\mathbf{\Sigma} = \mathbf{n}d\Sigma$  is the unit normal component of the surface element. See Fig. 2 for a schematic of the used notation. For more detailed information on different filter functions and their effect on the nature of the  $\mathbf{A}$  matrix, the readers are referred to [8].

As discussed, the key component required for any gradient-based shape optimization method is the objective and constraint function gradients with respect to the design variables  $\frac{d(\mathbf{J}, \mathbf{g}, \mathbf{h})}{ds}$  which will be used to generate the geometry update  $\Delta \mathbf{s}$ . In case, the derivatives of the functions  $\mathbf{J}, \mathbf{g}, \mathbf{h}$  are computed w.r.t to the spatial coordinates



**Fig. 2** Notional schematic of the design surface ( $\Gamma$ ), the filter function ( $F$ ), the integration area ( $\Sigma$ ) and the boundary of  $\Sigma$  ( $\Psi$ )

$\frac{d(J, g, h)}{dx_0}$ , it is required to compute the shape derivative, i.e., the derivative of the surface coordinates w.r.t. control field  $\frac{dX_0}{ds}$ , see Eq. 7. Various established methods for calculating the sensitivities are in practice; for example [5] shows a discrete adjoint method, [6] introduces a semi-analytical method, and [19] among others showcases a continuous adjoint method this adjoint sensitivity calculation is extended to unsteady problems in [13]. Some of these methods are already available for usage in commercial and open-source software packages. An in-depth discussion about the available methodologies for calculation of sensitivities is out of scope for the current contribution. Since the current contribution focuses on different aspects the framework necessary for shape optimization and the applications themselves, in the following discussion, we consider the shape sensitivities are given and the computational costs involved are not considered though they cannot be neglected. In the examples presented in Sect. 4, the methods used for calculating the sensitivities are explicitly mentioned for each application as multiple methods are used. This in turn establishes the versatility of the framework much strongly.

Given the control field and the objective gradients at the  $k^{th}$  optimization iteration, the new set of design variables can be calculated as:

$$s_{\Gamma}^{k+1} = s_{\Gamma}^k + \Delta s_{\Gamma}^k \tag{5}$$

where  $\Delta s_{\Gamma}^k$  is the update of control field. For a simple steepest descent algorithm, the computation of the new design can be written as:

$$s_{\Gamma}^{k+1} = s_{\Gamma}^k - \alpha \nabla s_{\Gamma}^k \tag{6}$$

Here  $\alpha$  is the optimization step length and  $\nabla s_{\Gamma}^k$  is the gradient of the objective  $J$  with respect to the control variables  $s$ . Other more advanced algorithms like sequential quadratic programming [20], front propagating algorithms [28] can also be used to generate the value of  $\Delta s_{\Gamma}^k$ . In the current contribution, unless otherwise mentioned, a steepest descent algorithm is used to generate the update of the control field.  $\nabla s_{\Gamma}^k$  can be calculated using the chain rule of differentiation. Thus, the objective derivative

with respect to the shape controls reads:

$$\frac{dJ}{ds_{\Gamma}^k} = \frac{dJ}{dX_{\Gamma}^k} \cdot \frac{dX_{\Gamma}^k}{ds_{\Gamma}^k} \quad (7)$$

where  $\frac{dJ}{dX_{\Gamma}^k}$  and  $\frac{dX_{\Gamma}^k}{ds_{\Gamma}^k}$  are the objective derivative with respect to mesh coordinates and shape derivatives of vertex morphing, respectively. From Eq. 4, the value of  $\frac{dX_{\Gamma}^k}{ds_{\Gamma}^k}$  can be given by the expression [4,8,9,24]:

$$\frac{dX_{\Gamma}^k}{ds_{\Gamma}^k} \approx A^k \quad (8)$$

Following the above derivations, the shape update can be computed as:

$$\Delta X_{\Gamma}^k = A^k(X_{\Gamma}^k) \cdot \Delta s_{\Gamma}^k \quad (9)$$

The mapping operation in Eq. 9 ensures the smoothness of the shape update calculated. Finally, it is worth to mention that vertex morphing can preserve the main features of the design surface. More precisely, it allows shape changes which are not affecting the “design character” aesthetic and geometrical features [8,23]. As mentioned previously,  $F$  and  $r$  are the design handles which control the feature preservation. A good example is the preservation of feature lines (sharp edges), which could be achieved by a proper choice of the filter radius  $r$  [23]. The reason for this is that all the features smaller than the radius  $r$  are only subject to a bulk and rigid motion, without considerable shape deformation. Another example would be the thickness control which results in the no self-penetration property.

For applying the constraints defined by  $\mathbf{g}$ ,  $\mathbf{h}$  their derivatives  $\frac{d(\mathbf{g}, \mathbf{h})}{dX_0}$  are transferred to the control space by the following operation

$$\frac{d(\mathbf{g}, \mathbf{h})}{ds_{\Gamma}^k} = A^k \frac{d(\mathbf{g}, \mathbf{h})}{dX_{\Gamma}^k} \quad (10)$$

and the constraints can be applied using one of the many methods available. In this work, a version of the gradient projection method introduced in [22, Chapter 5] is used to apply the constraints. This method satisfies constraints by projecting the descent direction onto the subspace tangent to the active constraints. Furthermore, a correction is performed to bring the design update back to the feasible domain.

As a conclusion of this section, Algorithm 1 outlines the steps followed and computations required for the constrained node-based shape optimization by means of vertex morphing.

**Algorithm 1** Active-set based constrained shape optimization using vertex morphing

---

```

//initialization
1:  $\mathbf{X}_\Gamma^0 \leftarrow$  discretize design surface  $\Gamma$ 
2:   discretize bounding surface  $\Gamma_b$ 
3:   search structures on  $\Gamma$  and  $\Gamma_b$ 
//optimization loop
4: for  $k = 1, 2, \dots$  do
// solve state problem using external solvers
5:    $\mathbf{r}^k(\mathbf{X}_\Gamma^k, \mathbf{Q}(\mathbf{X}_\Gamma^k)) = \mathbf{0}$ 
// calculate shape gradients of objective also using external solvers
6:    $\frac{dJ}{d(\mathbf{X}_\Gamma^k)}$ 
// find active inequality constraints
7:    $g_j^k \geq 0, j = 1, \dots, m_{ga}$ .
// calculate shape gradients of active constraints
8:    $\mathbf{C}^k \leftarrow \begin{bmatrix} \frac{d^k g_a}{d^k \mathbf{X}_\Gamma} \\ \frac{dh}{d^k \mathbf{X}_\Gamma} \end{bmatrix}_{(m_h+m_{ga}) \times 3m_\Gamma}$ 
// calculate mapping matrix
9:    $\mathbf{A}^k \leftarrow \frac{d(\mathbf{X}_{\Gamma,i}^k)}{d(s_\Gamma^k)} = \int_\Sigma F(\mathbf{X}, \mathbf{X}_i) d\Sigma, i = 1, \dots, m.$ 
// map gradients of objective to control space Eq. 7
10:   $\frac{dJ}{d(s_\Gamma^k)} \leftarrow \frac{dJ}{d(\mathbf{X}_\Gamma^k)} \cdot \mathbf{A}^k$ 
// map gradients of active constraints to control space
11:   $\mathbf{C}_m^k \leftarrow \mathbf{C}^k \cdot \mathbf{A}^k$ 
// calculate feasible search direction using algorithm of choice (Eq. 9, 6)
12:   $\mathbf{p}^{c,k} \leftarrow$  Calculate search direction
//calculate design update
13:   $\Delta s^k \leftarrow \alpha \mathbf{p}^{c,k}$ 
//map design update to geometry space
14:   $\Delta \mathbf{X}_\Gamma^k \leftarrow \mathbf{A}^k \cdot \Delta s_\Gamma^k$ 
15:  check convergence
16: end for

```

---

### 3 Software Framework

In the fast-evolving design process involving teams from different backgrounds, the ability to test various ideas with several constraints and objectives is fundamental. For optimization, this means the ability of the framework to achieve this with minimal effort from the user. For efficiently solving the optimization problem described in Algorithm 1, the following vital capabilities of the framework are noteworthy:

**Usage of external tools and solvers as black-box** In the optimization workflow described in Algorithm 1, the objective value and its gradient (statement 6), constraint values and their gradients (statement 8) with respect to the shape are calculated by the external solvers and provided to the optimization framework. This enables a modular and black-box treatment of the solvers for evaluating the responses and if necessary constraints. This approach also allows the framework to perform optimization based on different physics and design parameters. Based on this rationale, a black-box treatment



of the solvers is used in this work. For such a design, the communication module with different external solvers to transfer the quantities as mentioned earlier becomes a vital part of the optimization framework.

**Parametrization techniques** In shape optimization, parametrization of geometry is the procedure to define a set of parameters to control the shape that is optimized. Choice of a the parametrization can result either in a reduction or an increase in the number of design variables, thereby effecting the computational effort. Different parametrization techniques have been previously proposed and used [33]. As not every technique is suitable, efficient, and possible for every optimization problem, the optimization framework should have the ability to extend and assimilate different parametrization techniques quickly. The design structure of the framework allows for different parametrizations. The current work implements a discrete node-based parametrization approach described in Sect. 2.1. The versatility of the presented framework is also demonstrated in its ability to nest between two different parametrizations during the optimization processes. An example is presented in Sect. 4.1.

**Computing geometric and other specialized responses** As mentioned previously, the external physics solvers perform the calculation of state-based responses, constraints, and their gradients. In addition to such responses, other types like geometric [4], stamping constraints presented in Sect. 4.2.1 are also crucial in industrial applications of shape optimization. Computation of some of these specialized responses may not be possible with the external solver. In such cases, the possibility to compute them within the framework is necessary.

These features are critical in enabling the researchers to experiment with algorithmic procedures and allow the usage of the framework for industrial applications by practitioners.

### 3.1 Implementation

The implementation of the optimization framework is conceptually divided into two parts: first, the computation and memory intensive part, and second is the algorithmic part which organizes the data and workflow. Considering the nature of these two parts, C++ and Python are chosen, respectively, for them. This combination simultaneously enables efficient computations and improved readability of the algorithmic procedure. The Python layer also helps with fast and easy prototyping and setup of new and experimental optimization procedures.

Figure 3 shows the thematic arrangement of different components of the framework and steps in the optimization workflow. It also illustrates the critical aspects of the framework, of which the distinction between Python and C++ layers and the interface with the external solvers are noteworthy. The C++ implementation deals with computationally intensive tasks of formulating the mapping between different parametrizations (Eq. 1), node-search, linear algebra, and the data structure. The second part, implemented in Python, deals with orchestrating the functionalities in an optimization algorithm.

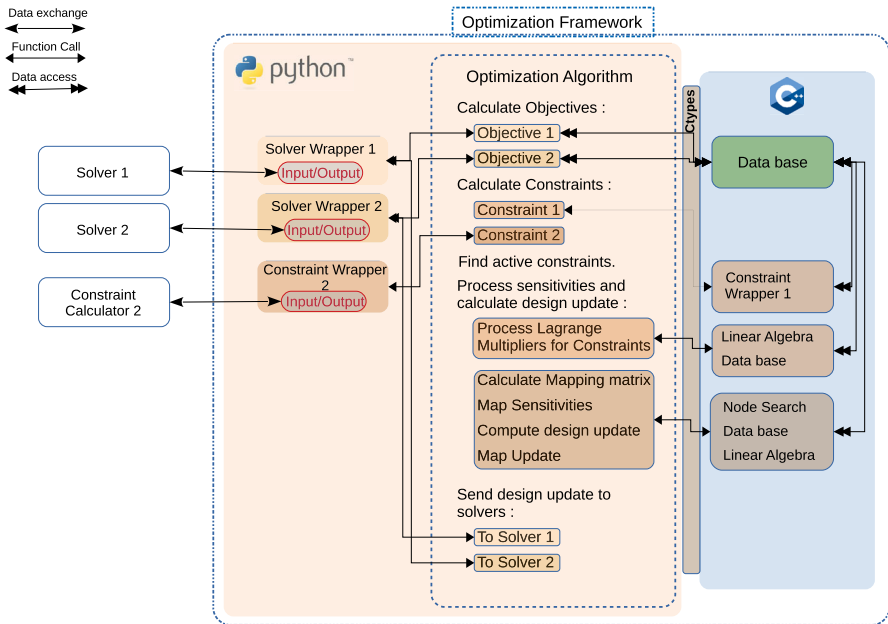


Fig. 3 Illustration of the optimization framework

**Parameterizations and mappers** In the vertex morphing technique described in Sect. 2, though the design parameters are the nodal coordinates of the discretized surface, the optimization happens on a different control space  $s$ . A transformation between the geometry coordinates and control parameters is performed using a mapping matrix (Eq. 8). This same procedure can be used for transformation between arbitrary parameterizations. Such generalization allows the possibility of having different parameterizations. Section 4.1 presents a series of nested parameterizations using this generalization.

**Communication with black-box solvers** One of the essential features of a general optimization framework is its ability to work with solvers treating them as “black-box” and communicate the objective and constraint values, their gradients and the shape updates. Such black-box treatment of solvers enables the framework to work on different optimization problems with different physics involved. A novel detached-interface approach enabling modularity with minimal and framework independent changes to the solver is developed. The features of this approach are:

- The interaction between the framework and the external “black-box” solver(s) is done via a solver-wrapper developed to adhere to the interface of the base solver in the framework. This allows uniform treatment of different solvers-wrappers and easy switch between them. Together with functions for exchanging data, this interface also contains functions to control the solver. A solver-wrapper will only implement the necessary functions depending the degree of control the “black-box” solver allows. This wrapper can also contain specific routines for communicating

1	<code>Initialize()</code>	8	<code>ImportModel()</code>
2	<code>Finalize()</code>	9	<code>ExportModel()</code>
3	<code>InitializeOptimizationStep()</code>	10	<code>ImportDataField()</code>
4	<code>FinalizeOptimizationStep()</code>	11	<code>ExportDataField()</code>
5	<code>CalculatePrimal()</code>	12	<code>SetIO()</code>
6	<code>CalculateResponseValue()</code>	13	<code>AddResponseFunction()</code>
7	<code>CalculateSensitivities()</code>		

**Fig. 4** List of functions in the solver-wrapper interface

the data with the actual solver. Figure 4 shows the important functions in the interface.

- The actual data communication between the framework is done via a different input/output (IO) class object. The functions from Fig. 4, `ImportDataField`, `ExportDataField`, `ImportModel`, `ExportModel` are responsible for the data exchange in the solver-wrapper. These functions are then delegated to the IO class object. This delegation of data exchange to an interfaced IO class enables decoupling the IO from the solver-wrapper and thus enables re-usage of existing IO methodologies and implementations between different solver-wrappers.
- The external “black-box” solver is completely independent from the developed framework. This implies that the solver can choose and implement any routines to export the objective or constraint sensitivities together with their values. This will enable reuse of already existing routines developed for other tools, thus simplifying the deployment of the solvers on different computing environments.

The following solvers-wrappers for external solvers are implemented in the framework using the above-described detached interface approach and are readily available for usage.

- OpenFOAM [37]
- SU2 [30]
- KratosMultiphysics [12,25]
- Altair OptiStruct
- StarCCM
- CARAT++ (*Internal structural analysis tool at Chair of Structural Analysis, TUM*)
- AVLFire
- Abacus

The usage of the above solver-wrappers is also stated in the numerical examples presented in Sect. 4. The following are the steps to setup a new solver-wrapper, using the above-described detached interface, and use a new black-box solver together with the optimization framework:

- Implement the following routines in the solver
  - To output the objectives and their sensitivities.
  - To accept the design update from the optimization framework.
  - To accept a convergence signal from the optimization framework.

This implementation in the solver has no dependency on the optimization framework whatsoever. Thus, the developers are free to use any tool of their choice and convenience to achieve this functionality.

- Use the interface provided by the optimization framework to create a wrapper to the solver either in C++ or in Python. This wrapper will be responsible for:
  - Interacting with the solver and give necessary instructions to the solver to perform different computations. For example, to compute the primal or sensitivity values.
  - Provide the design update computed by the optimization framework to the solver.

Though these operations depend upon the data structure of the optimization framework, these can be customized to the needs specific to the solver. For example, data transfer via file I/O or in memory transfer using MPI protocols.

The geometric constraints and the procedures presented in Sect. 4 are implemented following the procedure mentioned above and are included in the framework.

## 4 Optimal Shape Design Applications and Numerical Results

In the following section, we present different application cases which are optimized using the above-described optimization framework. Since the objective of the presented examples is to showcase the capabilities of the framework, description of the specialized optimization procedures adapted is provided when possible and references are provided otherwise. Unless explicitly mentioned, all the simulations below have a numerically stable behavior. The application examples are grouped in different focus areas, demonstrating capabilities of the framework and the special optimization procedures applied.

### 4.1 Focus Area: Multi-physics and Multi-objective Shape Optimization

This application example is a multi-physics problem from BMW Motorsport: Aerodynamic and structural shape design of the BMW M8 GTE wheel (Fig. 5). As can be seen in Fig. 6a, the spinning structure is put in a virtual wind tunnel to evaluate the aerodynamic performance. A moving reference frame (MRF) is used to account for the rotation of the wheel. A finite-volume discretization with 8 million cells forms the fluid domain and is modeled using in-compressible Navier–Stokes equations. The structural model is made up of approximately 1 million tetrahedral finite elements and rigid body elements. The fluid and structural models including boundary conditions are shown in Fig. 6.

OpenFOAM and Altair OptiStruct are used to solve the primal quantities fluid and structural models, respectively. The outer surface of the spokes is chosen as the design surface, which is also the interface between the fluid and the structure models. The surface shape sensitivities of the fluid are calculated using the adjoint solver of OpenFOAM: `adjointOptimizationFoam`, and the structural surface shape sensitivities are calculated using adjoint technology of OptiStruct. The surface mesh from fluid and structure do not match due to the different mesh requirements for the flow and structure. The structure interface mesh consists of 28,000 nodes, while the fluid inter-

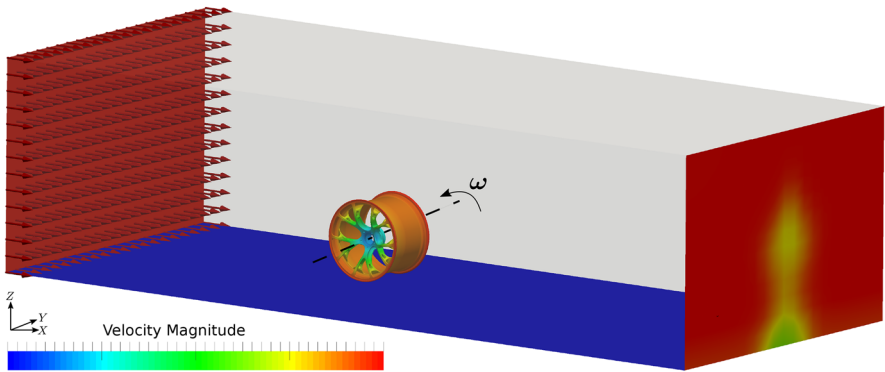


(a) BMW M8 GTE

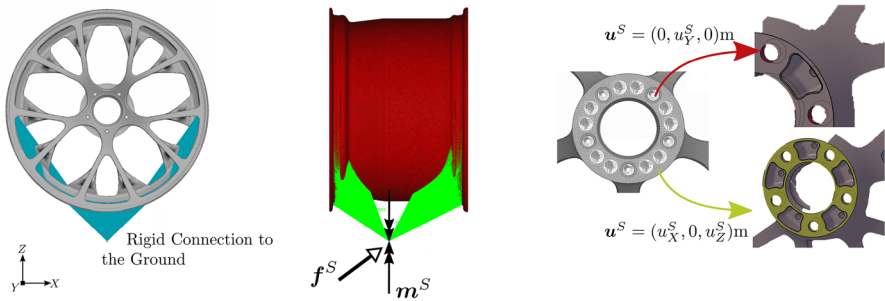


(b) Wheel.

Fig. 5 BMW M8 GTE and the corresponding wheel rim



(a) Fluid model.



(b) Structural model.

Fig. 6 Computational models of the rim

face mesh consists of 145,000 nodes. To deal with the non-matching meshes at the fluid–structure interface, we take advantage of the important property of the vertex morphing technique presented in Sect. 2.1 of being able to discretize the geometry and control spaces with different mesh resolutions. For this purpose, the interface mesh with less resolution, here the structure mesh with 84,000 design variables, should be chosen as the control mesh. The motivation of this choice comes from the fact that the representation of design modifications on the coarser mesh on a finer mesh is possible but not vice versa. Then, the objective functions’ gradients w.r.t. the design variables ( $\nabla_s J^F, \nabla_s J^S$ ) and the geometries’ updates ( $\Delta X_\Gamma^F, \Delta X_\Gamma^S$ ) are calculated, respectively, as

$$\frac{dJ^F}{ds_\Gamma^S} = \frac{dJ^F}{dX_\Gamma^F} \cdot \frac{dX_\Gamma^F}{ds_\Gamma^S} = \frac{dJ^F}{dX_\Gamma^F} \cdot [A^{FS}]_{3m_\Gamma^F \times 3m_\Gamma^S} \tag{11a}$$

$$\frac{dJ^S}{ds_\Gamma^S} = \frac{dJ^S}{dX_\Gamma^S} \cdot \frac{dX_\Gamma^S}{ds_\Gamma^S} = \frac{dJ^S}{dX_\Gamma^S} \cdot [A^{SS}]_{3m_\Gamma^S \times 3m_\Gamma^S} \tag{11b}$$

and

$$\Delta X_\Gamma^F = A^{FS} \cdot \Delta s_\Gamma^S \tag{12a}$$

$$\Delta X_\Gamma^S = A^{SS} \cdot \Delta s_\Gamma^S \tag{12b}$$

where  $J^F$  and  $J^S$  represent objective functions whose spatial gradients on the fluid surface mesh and the structure surface mesh, respectively. The matrices  $A^{FS}$  and  $A^{SS}$  are the operators which define the association between the non-matching interface meshes and the discretized design space. Based on the derivation of vertex morphing in Sect. 2.1, the entry in row  $i$  and column  $j$  of  $A^{FS}$  is computed as

$$A_{ij}^{FS} = \frac{dX_{\Gamma,i}^F}{ds_{\Gamma,j}^S} = \int_{\Sigma^S} F(X, X_{\Gamma,i}^F) d\Sigma \tag{13}$$

where  $\Sigma^S$  is the portion of  $\Gamma^S$  which lies within a sphere of radius  $r$  and center  $X_{\Gamma,i}^F$ .

Multiple response functions quantify the performance of the described problem. There are multiple ways to deal with multiple objectives in an optimization problem. Some of them include weighted-sum method, weighted min–max method, lexicographic and Pareto optimization methods. In the framework presented, owing to its highly effective and simple formulation, a weighted-sum method is used. In this method, a compromise function is formed as a weighted sum of existing objectives, the total drag force ( $J^F$ ) acting on the wheel, and the total strain energy ( $J^S$ ) of the wheel. This forms a new objective, thus transforming a multi-objective into a single-objective optimization problem. To give both the objectives equal importance in the optimization, equal weights are chosen for them in forming the combined objective. The following geometric constraints act on the shape optimization: (a) an equality constraint on the inner volume of the wheel (mass) and (b) an  $n$ -fold cyclic symmetry condition on the design surface. While the former is a single scalar-valued constraint,

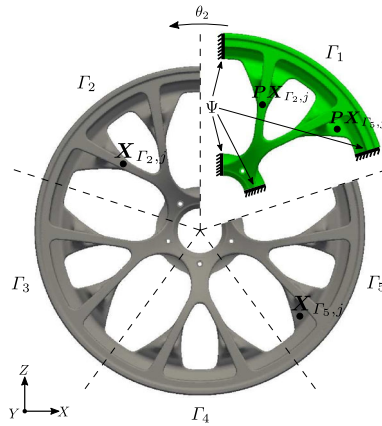


Fig. 7 Cyclic symmetry of the wheel and the design surface (Green)

the later results in numerous point-wise geometric constraints. Figure 7 explicitly focuses on the cyclic symmetry property of the wheel. As can be seen, the wheel surface  $\Gamma$  is divided into five identical surfaces  $\Gamma_i, i = 1, \dots, 5$ , each of which is generated by rotating  $\Gamma_1$  by  $\theta_i = \frac{(i-1)\pi}{5}$  around the Y-axis. In the current implementation, making the shape gradients rotationally symmetric enforces the cyclic symmetry constraint. The following linear transformation makes the shape gradients rotationally symmetric:

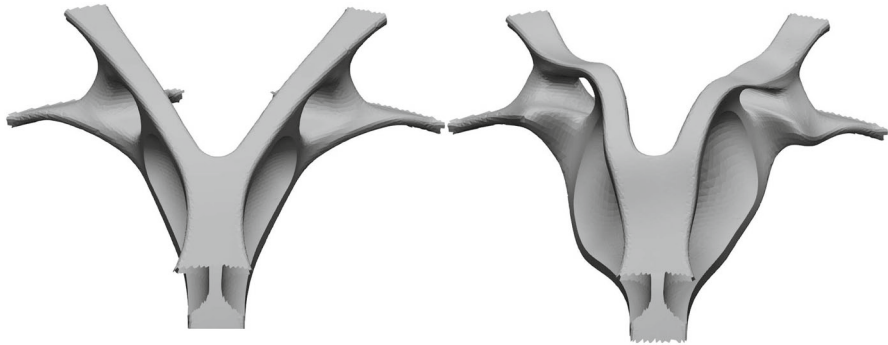
$$X_\Gamma = \begin{bmatrix} X_{\Gamma_1} \\ X_{\Gamma_2} \\ X_{\Gamma_3} \\ X_{\Gamma_4} \\ X_{\Gamma_5} \end{bmatrix} = \begin{bmatrix} [AP_{1,1}] \\ [AP_{2,1}] \\ [AP_{3,1}] \\ [AP_{4,1}] \\ [AP_{5,1}] \end{bmatrix} \cdot X_{\Gamma_1}, AP_{i,1} = \begin{bmatrix} T_i & 0 & \dots & 0 \\ 0 & T_i & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & T_i \end{bmatrix}_{3m_{\Gamma_i} \times 3m_{\Gamma_1}} \quad (14)$$

where  $AP_{i,1}, i = 1, \dots, 5$  is the transformation matrix from surface  $\Gamma_i$  to  $\Gamma_1$ .  $T_i$  is the rotation matrix for each section, and it may be calculated as

$$T_i = \begin{bmatrix} \cos \theta_i & 0 & -\sin \theta_i \\ 0 & 1 & 0 \\ \sin \theta_i & 0 & \cos \theta_i \end{bmatrix} \quad (15)$$

Then the shape variations in the surface  $\Gamma$  are associated with the shape variations of the surface  $\Gamma_1$  as follows:

$$\frac{dX_\Gamma}{dX_{\Gamma_1}} = AP \quad (16)$$



**Fig. 8** Initial versus optimized design of the section

**Table 1** Summary of the multi-objective optimization of the BMW M8 GTE wheel

Function	Change (%)
Total drag	– 35
Compliance 1	– 17.76
Compliance 2	– 16.38
Compliance 3	– 9.54
Compliance 4	– 19.47
Compliance 5	– 19.47
Compliance 6	– 19.40

Finally, the rotationally symmetric form of objectives’ gradients can be calculated by using the following equations:

$$\left. \frac{dJ^F}{ds_\Gamma^S} \right|_{rs} = \frac{dJ^F}{dX_\Gamma^F} \cdot AP^F \cdot (AP^F)^T \cdot A^{FS}, \quad AP^F = \frac{dX_\Gamma^F}{dX_{\Gamma_1}^F} \tag{17}$$

$$\left. \frac{dJ^S}{ds_\Gamma^S} \right|_{rs} = \frac{dJ^S}{dX_\Gamma^S} \cdot AP^S \cdot (AP^S)^T \cdot A^{SS}, \quad AP^S = \frac{dX_\Gamma^S}{dX_{\Gamma_1}^S} \tag{18}$$

The above-described process of making the surface gradients symmetric is implemented in the current frame work taking advantage of its modular nature of implementing at C++ level shown in Fig. 3. Table 1 and Fig. 8 summarizes the outcome of the described optimization problem. It shows a clear improvement in drag and compliance. One can also notice the cyclic symmetry in the resultant design geometry presented in Fig. 9. This example, in addition to demonstrating the ability to work with different solvers for physics and using multiple objective functions, also demonstrates the possibility to introduce geometrical constraints into the parametrization.



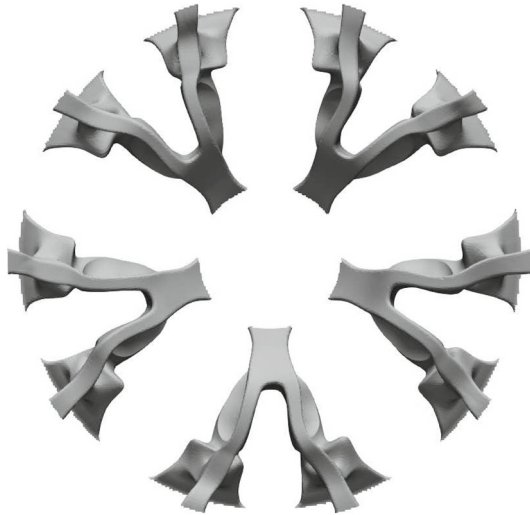


Fig. 9 Optimized shape and the final improvement in the compliance

## 4.2 Focus Area: Flexibility and Adaptability for Manufacturing Constraints

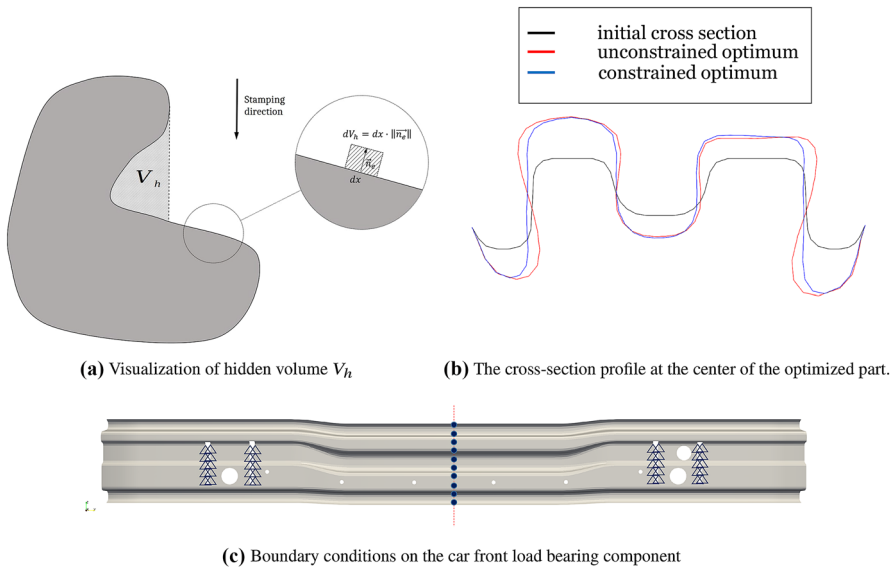
Real-world industrial applications rarely produce unconstrained shape optimization problems. Physical and geometric constraints arise naturally due to the limitations of the physical properties or those of the manufacturing process of the mechanical component. In such applications, individual manufacturing requirements have to be fulfilled by the optimized geometry. Those are relevant to the respective manufacturing process, such as casting, molding and stamping. In the newly emerging fields like 3D printing, new types of constraints arise on the shape of the 3D printed parts.

These different types of constraints and their inclusion in the optimization process requires a flexible framework. In the following, we present two examples of such specialized constraints implemented in the current framework to demonstrate its the flexibility and adaptability.

### 4.2.1 Stamping

This example presents the inclusion of the stamping constraint. The optimization problem consists of a load-bearing member of a car subjected to respective load conditions, as shown in Fig. 10c. The member is shape-optimized for minimal strain energy. In addition to a mass constraint, the stamping constraint is applied to the optimization problem. Considering the surface of the component as design geometry, Altair OptiStruct is used for structural calculation of primal and shape sensitivities. A mesh of 10500 nodes is used for the computation.

There are multiple ways to impose the stamping manufacturability constraint. This framework approaches the problem from a geometric point of view. The following condition describes a perfectly stampable geometry: for every node, when a ray trace is shot from it in the stamping direction, the ray should not collide with any element



**Fig. 10** Stamping test case

of the mesh. In other words, looking at the mesh in the stamping direction, all nodes should be visible. The volume  $V_h$  formed by the hidden parts of the mesh forms the constraint objective. Such a constraint is called a soft constraint as it is not satisfied node wise but on average. Figure 10a shows the hidden volume  $V_h$  visually. Numerically,  $V_h$  is calculated using ray-tracing. A novel radius search in the mesh is employed to reduce the computational costs induced by ray tracing. Figure 10b shows the optimized cross section of the component with and without stamping constraints. The described geometric constraint is specifically designed to ensure that the optimized geometry can be produced using a stamping manufacturing process. Here, the geometric formulation replaces a much more elaborate simulation of the manufacturing process. This application case shows the ease of integration of such specific constraint formulations into the optimization process.

#### 4.2.2 Stackability

The emerging and established techniques in the field of additive manufacturing and 3D printing impose a vital requirement of stackability on the components being printed. This requirement is more pronounced when using powder bed-based techniques in metal printing. A stackable component allows printing of more components per unit area. This transforms into a geometrical requirement of placing two consecutive parts as close as possible to each other. In the current framework, this is realized as a geometric no penetration constraint. The current framework operating at BMW Group employs this technique successfully to different 3D-printed geometries.

An exemplary application is presented here. It consists of a structural component subjected to pseudo-load conditions. As shown in Fig. 11a, the component is to be

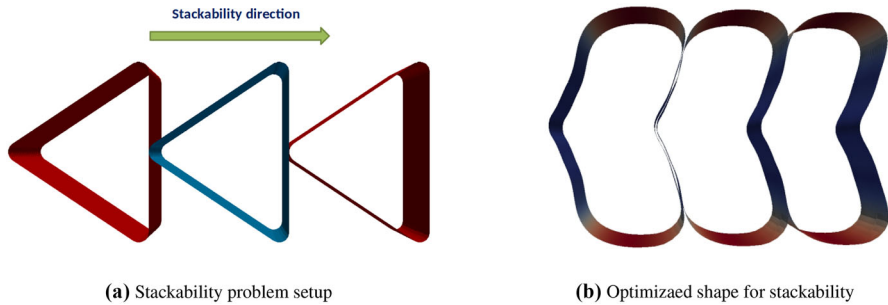


Fig. 11 Stackability test case

stackable in the given direction. Altair OptiStruct is used for simulation of the structural model with 22,000 nodes. The shape of the component is optimized for minimal strain energy subjected to constraints on stackability. The stackability constraint is implemented as a solver using the solver interface “SolverWrapper” of the framework. This solver instead of solving for the primal values will only perform the necessary geometric operations required for calculating the sensitivities for the constraint. Figure 11b shows the optimized component after 20 optimization iterations. As seen here, the shape of the component respects the no penetration with the two adjacent components and thus is more suitable for mass production using 3D printing technologies.

This application uniquely showcases the re-usability of existing features, in this case, geometric no penetration constraints, to formulate and realize new optimization problems to include diverse application fields.

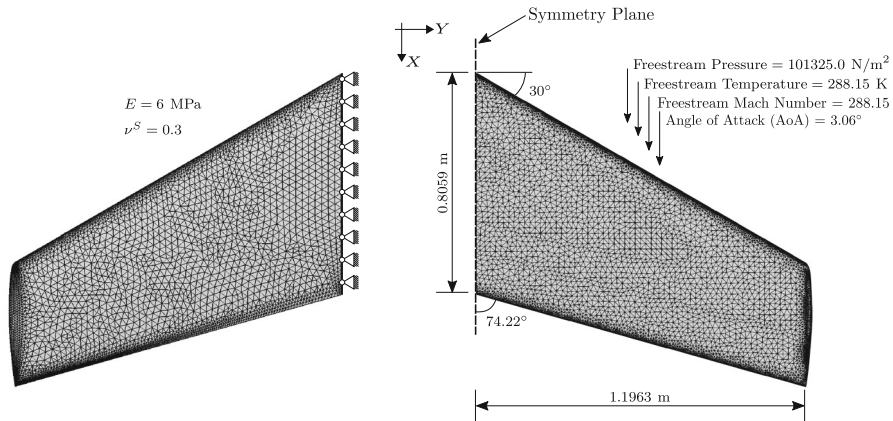
#### 4.3 Focus Area: Communication and Usage of Multiple Solvers

Multi-physics simulations have been an essential tool in understanding complex physical phenomena. In this context, multi-physics optimization is also equally important. An effortless way for performing such optimization is to use the existing primal solvers for calculations. This approach not only is modular but reuses the expertise of physical solvers independent of each other without introducing more complexity.

These multi-physics optimization problems introduce a new level of complexity to the enabling framework as these problems need to work with multiple solvers simultaneously and choreograph them in an optimization algorithm. The following two application cases make evident the capability of the software design of the optimization framework to work with such complex problems.

##### 4.3.1 FSI Shape Optimization: Flexible ONERA M6 Wing

The first example in this focus area is a multi-objective and multi-disciplinary shape optimization of a flexible ONERA M6 wing immersed in a compressible inviscid fluid flow. Both the fluid analysis (CFD) and the structural analysis (CSD) assume steady cruise conditions. For the CFD analysis, the compressible solver from SU2 [30] is used and the CSD analysis is carried out in KratosMultiphysics framework



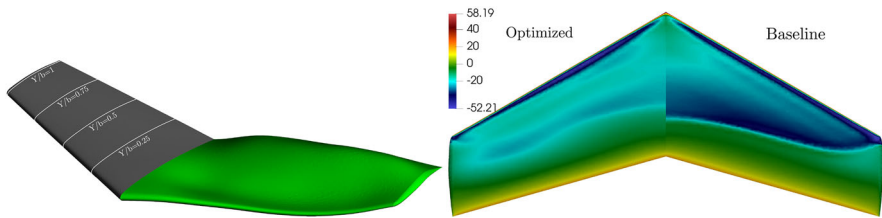
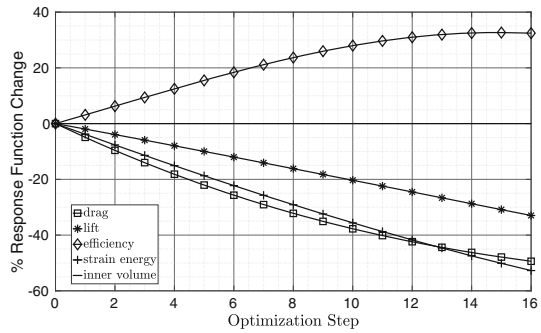
**Fig. 12** Description and surface discretization of ONERA M6 for FSI. Left: structural model, right: fluid model

[25]. The wing structure is clamped at the wing root, as shown in Fig. 12. The steady-state transonic flow over the ONERA M6 wing at Mach 0.8395 and angle of attack of  $3.06^\circ$  is computed using nonlinear Euler equations. The wing structure is purposefully modeled to produce large deformations and uses 4-node tetrahedral nonlinear solid elements. The flexible structure for the wing introduces fluid–structure interaction into the model. So the corresponding shape sensitivity analysis becomes an aeroelastic problem and requires a coupled sensitivity analysis to be performed. The optimization problem is formulated with the lift to drag ratio (efficiency) from fluid and strain energy as a weighted objective and the inner volume of the wing constraints this optimization problem. The shape sensitivities for these objectives are also calculated in the respective solvers. Naturally, the fluid and the structural domains have two different levels of refinement. As in the previous application of Sect. 4.1, the coarser wing surface mesh (structure surface mesh) is used to parametrize the surface using vertex morphing technique. Therefore, the filtering operations presented in Eqs. 11 and 12 are used in the optimization process.

The optimization resulted in a 32.4% increase in the lift to drag ratio and a 52% decrease in the total structural strain energy. The optimization history is presented in Fig. 13. Furthermore, Fig. 14a compares the optimized (scaled deformations) and initial configuration of the wing. As seen in Fig. 14b, the strong shock wave that existed along the span is reduced significantly. A detailed discussion about the methodology followed and results is presented in a parallel publication by one of the authors [3].

This example shows the applicability of the framework even for highly complex shape optimization with a fully coupled multi-physics problems involved. The shape update of the non-matching computational meshes is controlled using the vertex morphing method without additional modeling effort.

**Fig. 13** Optimization history for flexible ONERA M6 wing



(a) Left: baseline design, Right: final design scaled by 100 for better visualization.

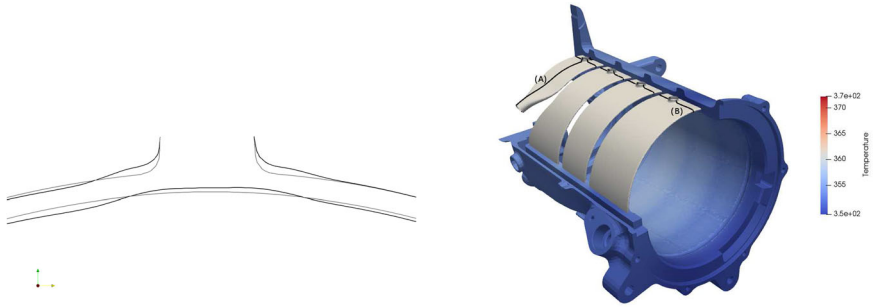
(b) Surface traction field (kPa) of the upper surface.

**Fig. 14** Results of shape optimization of ONERA M6 wing

### 4.3.2 Multi-disciplinary Shape Optimization: Spiral Water Jacket

Further expanding upon the use of vertex morphing in multi-disciplinary applications, the optimization of a spiral water jacket is investigated. Optimal design of an electric motor water jacket, seen in Fig. 15b, is a de facto multi-disciplinary problem requiring the consideration of many different physics. The primary role of the water jacket is to cool the motor. This cooling effect in turn improves power output and the life of components in and around the motor. The initial geometry, as shown in Fig. 15b, represents an internal flow region within the motor housing. This application of shape optimization with vertex morphing considers a simplified optimization considering only the flow and static structural properties. The objective of the optimization is to minimize the pressure drop while respecting a maximum stress constraint. Besides, an internal geometric restriction was placed on the geometry to maintain a minimum internal wall thickness between the water jacket and the stator located inside the house as previous runs had shown this to be a problem.

The flow and thermal problem are solved using a monolithic conjugate heat transfer (CHT) problem using the commercial solver Star-CCM+, while the structural analysis is carried out in the commercial solver Optistruct. The consideration of the thermal load in the structural solver was the source of the coupling between the models. The temperature field calculated in Star-CCM+ is applied to the Optistruct model via a mapping operation also implemented in optimization framework described. The design surface is the surface of the water jacket and the 22400 nodal coordinates on this surface act as control variables.



(a) Shape optimized core support from plane A. Gray is original and black is the optimized shape after 8 optimization steps.

(b) Original cut section of the housing geometry showing the water jacket and the selected planes.

Fig. 15 Optimized geometry for the core supports

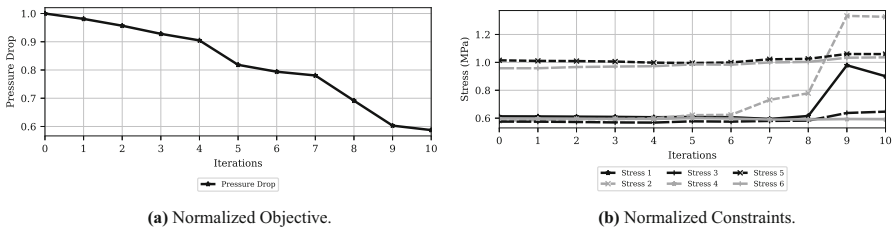
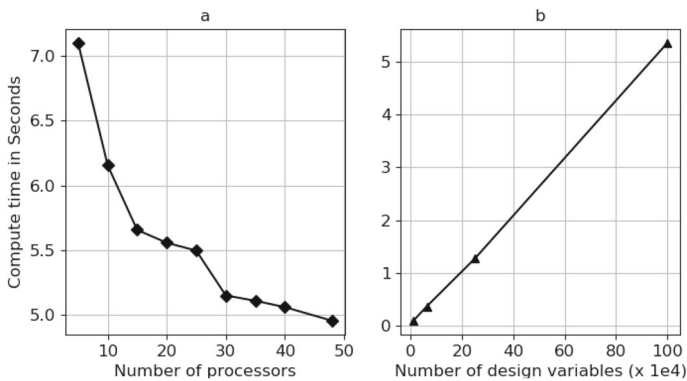


Fig. 16 Response functions from the optimization of the water jacket

The optimization was carried out for 10 steps before it stops because of mesh-related issues. Though geometric changes are small, the improvement in the pressure drop amounted to a reduction of about 30%. The main changes to the geometry are around the sand core support geometries. This is shown in Fig. 15a. These features exist due to the sand core casting technique used to manufacture the part. Although there are improvements in this case, a proper adjoint coupled problem may be more successful at maintaining stresses below the permissible limit. Additionally, further constraints and objectives representing the manufacturing technique may be required as the changes seen here are likely to have a direct impact on the manufacturing process as well. In this application case, in addition to the communication with two external solvers, the transfer of the temperature field from fluid to structural models is also performed using the current framework. This application is a prime example of the flexibility and generality of the interfaces and the implementation.

### 4.4 Computational Performance

Quantification of the computational performance of an optimization process is not straightforward as the components that consume the lion’s share of the computation time are the individual physics solvers computing the primal, possible response gradients. A discussion on performance of such solvers is a discipline on its own and is out of scope of current contribution as they are treated as black-boxes. Therefore,



**Fig. 17** Scaling for vertex morphing in the optimization framework

this subsection focuses on the performance of the framework itself; this includes the operations of 8 till 15 of Algorithm 1. To this extent, an exemplary test case of a flat square surface with pseudo-sensitivity information is taken to demonstrate the computational capabilities of the optimization framework. An additional factor that strongly influences the performance of the implemented vertex morphing technique is the filter radius illustrated in Fig. 2. This determines the average number of neighbors, for each node, which are inside the radius. Figure 17a shows the strong scaling plot of the framework while keeping the average number of neighbors constant. Figure 17b also shows the performance of the framework for the case with 1,000,000 nodes and an increasing number of neighbors. As expected, the computational cost increases as the number of neighbors increases, but only linearly. This shows that the framework is efficient in both regards.

## 5 Conclusion and Outlook

In this work, we introduced a concept of a framework for performing shape optimization using vertex morphing. The framework offers a flexible interface to use different external physics solvers. For this, an innovative detached-interface approach is introduced. Multiple commercial and open-source tools used in the applications give an insight into the generality of this approach. It also has the possibility to include geometric and other special constraints in the optimization process. Wide range of application cases with different levels of the complexity, especially involving multi-physics simulations together with geometrical constraints, prove the flexibility and robustness of the framework presented. This design of the framework drawing on the object-oriented features of C++, and the scripting nature of Python programming languages provide a high degree of flexibility for the developers and practitioners to realize new optimization problems. The efficiency and the capability of the framework in handling a large number of design variables are also showcased with the applications.

As an outlook, coupled adjoint sensitivities are planned be used in the example presented in Sect. 4.3.2 to achieve more improvement in the objective. A constant



effort in including more commercial solver is done to increase the number of applications. Further additions to the specific industrial production constraints are in progress together with industry partners.

**Acknowledgements** This paper is a part of the research project sponsored by the BMW group, Munich. The first two authors gratefully acknowledge support by Deutsche Forschungsgemeinschaft (DFG) through TUM International Graduate School of Science and Engineering (IGSSE), GSC 81 under projects 12.01 and 9.10. The authors also gratefully acknowledge the guidance, inputs and support of Dr.-Ing. Majid Hojjat of BMW Group, Munich. The authors acknowledge the support of colleagues Mr. Daniel Baumgartner for his inputs during the initial design and development of the optimization framework and Mr. Ihar Antonau who continue to actively use and develop the optimization framework. The authors are grateful to Dr. Steffen Jahnke, Stefanus Stahl and Michael Hofer of BMW Group, Munich, for providing high-fidelity models for optimization. The authors also express gratitude to their colleague Mr. Shahrokh Shayegan for his valuable insights. We are also thankful for the whole optimization framework team at BMW Group, Munich, for their co-operation and support during the preparation of the manuscript. A part of the results discussed here in this work are presented in International Conference on Simulation for Additive Manufacturing-Sim-AM 2019 [18].

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Data Availability Statement** The data and the numerical results presented in this work are available to interested readers on reasonable request to the authors.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Aage, N., Andreassen, E., Lazarov, B.S., Sigmund, O.: Giga-voxel computational morphogenesis for structural design. *Nature* **550**(7674), 84–86 (2017). <https://doi.org/10.1038/nature23911>
2. Alexandersen, J., Sigmund, O., Aage, N.: Large scale three-dimensional topology optimisation of heat sinks cooled by natural convection. *Int. J. Heat Mass Transf.* **100**, 876–891 (2016). <https://doi.org/10.1016/j.ijheatmasstransfer.2016.05.013>
3. Asl, R.N.: Shape optimization and sensitivity analysis of fluids, structures, and their interaction using vertex morphing parametrization. Ph.D. thesis, Technische Universität München, München. <https://mediatum.ub.tum.de/doc/1487664/1487664.pdf> (2019)
4. Asl, R.N., Shayegan, S., Geiser, A., Hojjat, M., Bletzinger, K.U.: A consistent formulation for imposing packaging constraints in shape optimization using Vertex Morphing parametrization. *Struct. Multidiscip. Optim.* (2017). <https://doi.org/10.1007/s00158-017-1819-9>
5. Balasubramanian, R., Newman III, J.C.: Discrete direct and adjoint sensitivity analysis for arbitrary mach number flows. *Int. J. Numer. Methods Eng.* **66**(2), 297–318 (2006). <https://doi.org/10.1002/nme.1558>
6. Barthelemy, B., Haftka, R.T.: Accuracy analysis of the semi-analytical method for shape sensitivity calculation. *Mech. Struct. Mach.* **18**(3), 407–432 (1990). <https://doi.org/10.1080/08905459008915677>
7. Bertsekas, D.P.: On the Goldstein–Levitin–Polyak gradient projection method. *IEEE Trans. Autom. Control* **21**(2), 174–184 (1976). <https://doi.org/10.1109/tac.1976.1101194>
8. Bletzinger, K.U.: A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape. *Struct. Multidiscip. Optim.* **49**(6), 873–895 (2014). <https://doi.org/10.1007/s00158-013-1031-5>



9. Bletzinger, K.U.: Shape optimization. In: Stein, E., de Borst, R., Hughes, T. (eds.) *Encyclopedia of Computational Mechanics*, Volume Set, vol. 2, 3, 2nd edn. Wiley, Hoboken (2017)
10. Bungartz, H.J., Lindner, F., Gatzhammer, B., Mehl, M., Scheufele, K., Shukaev, A., Uekermann, B.: preCICE—a fully parallel library for multi-physics surface coupling. *Comput. Fluids* **141**, 250–258 (2016). <https://doi.org/10.1016/j.compfluid.2016.04.003>. (**Advances in fluid–structure interaction**)
11. Chen, L., Bletzinger, K.U., Geiser, A., Wüchner, R.: A modified search direction method for inequality constrained optimization problems using the singular-value decomposition of normalized response gradients. *Struct. Multidiscip. Optim.* (2019). <https://doi.org/10.1007/s00158-019-02320-9>
12. Dadvand, P., Rossi, R., Oñate, E.: An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Arch. Comput. Methods Eng.* **17**(3), 253–297 (2010). <https://doi.org/10.1007/s11831-010-9045-2>
13. Economou, T., Palacios, F., Alonso, J.: An unsteady continuous adjoint approach for aerodynamic design on dynamic meshes. *AIAA J.* **53**(9), 2437–2453 (2014). <https://doi.org/10.2514/6.2014-2300>
14. Eldred, M., Dalbey, K., Bohnhoff, W., Adams, B., Swiler, L., Hough, P., Gay, D., Eddy, J., Haskell, K.: Dakota: a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Version 5.0, user's manual (2009). <https://doi.org/10.2172/991842>
15. Eschenauer, H.A.: Shape optimization of satellite tanks for minimum weight and maximum storage capacity. *Struct. Optim.* **1**(3), 171–180 (1989). <https://doi.org/10.1007/BF01637337>
16. Gallagher, R., Zienkiewicz, O.: *Optimum Structural Design: Theory and Applications: Based on a Series of Lectures Given at a Symposium Held in Swansea in January 1972*. Wiley. <https://books.google.de/books?id=nqIIswEACAAJ> (1973)
17. Gallard, F., Vanaret, C., Guenot, D., Gachelin, V., Lafage, R., Pauwels, B., Barjhoux, P.J., Gazaix, A.: GEMS: A Python Library for Automation of Multidisciplinary Design Optimization Process Generation. *AIAA SciTech Forum*. American Institute of Aeronautics and Astronautics (2018). <https://doi.org/10.2514/6.2018-0657>
18. Ghantasala, A., Asl, R.N., Stahl, S., Shayegan, S., Hojjat, M., Bletzinger, K.U.: Node based shape optimization for higher productivity in additive manufacturing. In: *II International Conference on Simulation for Additive Manufacturing—Sim-AM 2019*. Pavia, Italy (2019)
19. Giannakoglou, K.C., Papadimitriou, D.I.: Adjoint methods for shape optimization. In: *Optimization and Computational Fluid Dynamics*, pp. 79–108. Springer, Berlin Heidelberg (2008). [https://doi.org/10.1007/978-3-540-72153-6\\_4](https://doi.org/10.1007/978-3-540-72153-6_4)
20. Gill, P.E., Murray, W., Saunders, M.A.: Snopt: an sqp algorithm for large-scale constrained optimization. *SIAM Rev.* **47**(1), 99–131 (2005). <https://doi.org/10.1137/S003614450446096>
21. Gray, J., Hwang, J., Martins, J., Moore, K., Naylor, B.: Openmdao: an open-source framework for multidisciplinary design, analysis, and optimization. *Struct. Multidiscip. Optim.* (2019). <https://doi.org/10.1007/s00158-019-02211-z>
22. Haftka, R.T., Gürdal, Z.: *Elements of Structural Optimization*. Springer Netherlands (1992). <https://doi.org/10.1007/978-94-011-2550-5>
23. Hojjat, M.: Node-based parametrization for shape optimal design. Ph.D. thesis, Technische Universität München. <https://mediatum.ub.tum.de/doc/1231550/1231550.pdf> (2014)
24. Hojjat, M., Stavropoulou, E., Bletzinger, K.U.: The Vertex Morphing method for node-based shape optimization. *Comput. Methods Appl. Mech. Eng.* **268**, 494–513 (2014). <https://doi.org/10.1016/j.cma.2013.10.015>
25. KratosMultiphysics: <https://github.com/KratosMultiphysics/Kratos>. Accessed 26 Feb 2021
26. Le, C., Bruns, T., Tortorelli, D.: A gradient-based, parameter-free approach to shape optimization. *Comput. Methods Appl. Mech. Eng.* **200**(9–12), 985–996 (2011). <https://doi.org/10.1016/j.cma.2010.10.004>
27. Nonogawa, M., Takeuchi, K., Azegami, H.: Shape optimization of running shoes with desired deformation properties. *Struct. Multidiscip. Optim.* **62**(3), 1535–1546 (2020). <https://doi.org/10.1007/s00158-020-02560-0>
28. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.* **79**(1), 12–49 (1988). [https://doi.org/10.1016/0021-9991\(88\)90002-2](https://doi.org/10.1016/0021-9991(88)90002-2)
29. Othmer, C.: Adjoint methods for car aerodynamics. *J. Math. Ind.* **4**(1), 6 (2014). <https://doi.org/10.1186/2190-5983-4-6>

30. Palacios, F., Economou, T.D., Aranake, A.C., Copeland, S.R., Lonkar, A.K., Lukaczyk, T.W., Manosalvas, D.E., Naik, K.R., Tracey, B., Variyar, A., Alonso, J.J.: Stanford University Unstructured (SU 2 ): Open-source Analysis and Design Technology for Turbulent Flows (January), pp. 1–33 (2014)
31. Perez, R.E., Jansen, P.W., Martins, J.R.R.A.: pyopt: a python-based object-oriented framework for nonlinear constrained optimization. *Struct. Multidiscip. Optim.* **45**(1), 101–118 (2012). <https://doi.org/10.1007/s00158-011-0666-3>
32. Qin, H., Liu, Z., Liu, Y., Zhong, H.: An object-oriented matlab toolbox for automotive body conceptual design using distributed parallel optimization. *Adv. Eng. Softw.* **106**, 19–32 (2017). <https://doi.org/10.1016/j.advengsoft.2017.01.003>
33. Samareh, J.A.: Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA J.* **39**(5), 877–884 (2001). <https://doi.org/10.2514/2.1391>
34. Schmitt, O., Friederich, J., Riehl, S., Steinmann, P.: On the formulation and implementation of geometric and manufacturing constraints in node-based shape optimization. *Struct. Multidiscip. Optim.* **53**(4), 881–892 (2016). <https://doi.org/10.1007/s00158-015-1359-0>
35. Stavropoulou, E.: Sensitivity analysis and regularization for shape optimization of coupled problems. Ph.D. thesis, Technische Universität München, München. <https://mediatum.ub.tum.de/doc/1231547/1231547.pdf> (2015)
36. Wang, T.: Development of co-simulation environment and mapping algorithms. Ph.D. thesis, Technische Universität München, München. <https://mediatum.ub.tum.de/doc/1281102/1281102.pdf> (2016)
37. Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object-oriented techniques. *Comput. Phys.* **12**(6), 620–631 (1998). <https://doi.org/10.1063/1.168744>
38. Zavala, G.R., Nebro, A.J., Durillo, J.J., Luna, F.: Integrating a multi-objective optimization framework into a structural design software. *Adv. Eng. Softw.* **76**, 161–170 (2014). <https://doi.org/10.1016/j.advengsoft.2014.07.002>
39. Zhu, J.H., Zhang, W.H., Xia, L.: Topology optimization in aircraft and aerospace structures design. *Arch. Comput. Methods Eng.* **23**(4), 595–622 (2016). <https://doi.org/10.1007/s11831-015-9151-2>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.