48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition
4 - 7 January 2010, Orlando, Florida

AIAA 2010-1508

A Knowledge-Based Geometry Repair System for Robust Parametric CAD Models

Dong Li,[*] András Sóbester [†] and Andy J. Keane[‡]

University of Southampton, Southampton, Hampshire, SO171BJ, UK

In modern multi-objective design optimization (MDO) an effective geometry engine is becoming an essential tool and its performance has a significant impact on the entire MDO process. Building a parametric geometry requires difficult compromises between the conflicting goals of robustness and flexibility. This article presents a method of improving the robustness of parametric geometry models by capturing and modeling engineering knowledge with a support vector regression surrogate, and deploying it automatically for the search of a more robust design alternative while trying to maintain the original design intent. Design engineers are given the opportunity to choose from a range of optimized designs that balance the 'health' of the repaired geometry and the original design intent. The prototype system is tested on a 2D intake design repair example and shows the potential to reduce the reliance on human design experts in the conceptual design phase and improve the stability of the optimization cycle. It also helps speed up the design process by reducing the time and computational power that could be wasted on flawed geometries or frequent human interventions.

I. Introduction

The practice of engineering design optimization has gradually evolved from a manually, time consuming, step-by-step approach to an automated optimization process.[1] The automated frameworks that have been developed are abundant in literature and diverse in nature depending on the problem at hand. Nevertheless, most of them make use of a common component—a parametric geometry engine which generates parameterized geometrical models which are entirely defined by a set of design variables. The models serve as the starting point for subsequent analysis and evaluation, such as computational fluid dynamics (CFD) or computational structural mechanics (CSM), the analysis outcome(s) being sent back into the optimization framework, creating a design-evaluate-redesign workflow. In an ideal workflow, the geometry engine is able to deliver a geometry model defined by the set of design variables as and when required by the optimizer.

Because of the highly global nature of conceptual design search, the geometry engine in the optimization framework should be able to deliver a variety of different geometries defined by a wide range of design variable configurations without difficulty, i.e. the geometry engine should be flexible as well as robust. However, although parameterization technology has been a research focus for at least 20 years and various parameterization technologies having been developed,[2] the control of the trade-off between the desire for robustness and the need for flexibility is still a pressing challenge of parametric geometry generation. An expediential measure for ensuring robustness is to place tight bound limits to design variables so that any combination of the variables in the trimmed design space leads to a feasible design. However, for complex geometry models, the infeasible regions often exhibit irregular shapes, and are therefore hard to avoid. The above measure will then lead to either a very limited design space that to be explored, or some remaining infeasible region(s) which will make the model generation process fail from time to time.

Up to now, there is no satisfactory solution to the above problem. Especially for general-purpose commercial CAD packages, a flawless coverage of the design space is very difficult to realize. As a result, bespoke in-house geometry engines still dominate in the conceptual design phase. These bespoke engines are made

[*]Graduate Research Student, Computational Engineering and Design Group, School of Engineering Sciences.
[†]Lecturer, Computational Engineering and Design Group, School of Engineering Sciences, AIAA member.
[‡]Professor of Computational Engineering, School of Engineering Sciences.

specifically, according to the needs of an individual customer or product. They are usually more time-consuming, difficult and costly to build and use than off-the-shelf commercial CAD tools. Their applications are usually limited for specific problems, and within a company or an organization. Furthermore, there is no mechanism to guarantee that the precious engineering experience and knowledge which is used in the construction of a bespoke geometry engine can be preserved and further reused. So far, there is relatively little work that has addressed this problem.

In this work, we propose an automatic geometry repair system to handle the above problem. The system aims to repair geometrically or physically flawed geometries based on an engineering knowledge base, and therefore to assist the geometry engine to generate robust models without limiting its flexibility. The system aims to reduce the reliance on human design experts in the conceptual design phase and improve the stability of the optimization cycle. It also helps speed up the design process by reducing the time and computational power that could be wasted on flawed geometries or frequent human interventions. The prototype system aims to provide the following capabilities: capturing and storing the knowledge of a design engineer; synthesizing the knowledge into a general knowledge base; deploying the knowledge automatically to recommend a repaired geometry alternative when and as required; producing inferences that the human expert may not be able to devise in a reasonable amount of time. In a nutshell, the system aims to be a valuable tool in the automated design optimization.

It should be noted that the automated design optimization process could be further hindered by a range of other problems, for example, some geometric features or topological errors could not be automatically dealt with by CFD or CSM grid generation algorithms. This type of error is not strictly related to the parameterization of the geometry and often unrelated to the engineer's design knowledge. Therefore, they lie outside of the repair capability of the prototype system that is proposed in this paper. However, in the literature, the practice of preparing geometries for subsequent manipulation such as meshing is usually referred to as "geometry clean-up". It should not be confused with the term 'geometry repair' that is used here which refers to the automatic repair of infeasible designs in terms of its design variables.

In the next section a knowledge-based repair system is proposed and in Section III a case study is given.

II. A Knowledge-Based Geometry Repair System

A. Knowledge representation

An engineer's judgement and expert knowledge of the feasibility of a model can be drawn from various sources such as:

- explicit rules, discovered by using engineering or geometrical judgment; these rules include equality and inequality constraints, parameters and engineering laws, etc.,
- assessment of individual design cases by an expert engineer, and
- computational analysis results (from CFD, FEA, etc.).

There are no universally applicable schemes for incorporating engineering knowledge into a design system. In this work, we show the possibility of transferring some knowledge into explicit functions and incorporate the knowledge into a regression model. Furthermore, knowledge of specific designs take directly from individual engineers, existing designs or computational analysis results can be mapped to design variable sets.[3] The explicit rules can be transformed into penalty functions. The penalty function method is commonly used in design optimization to transform engineering constraints into objective functions, onto which various optimization techniques can then be applied. The simplest penalty approach in the optimization literature is to attach a large penalty constant to the original objective function whenever any constraint is violated. Although simple to apply, the approach causes a discontinuity in the shape of the penalized objective at the constraint boundary and takes no account of the number of constraint violations. These limitations can be mitigated by some modifications. First, the penalty can be multiplied by the degree of violation of the constraint for continuity. Secondly, a separate penalty function may be applied to each violated constraint. In our work, both modifications are adopted and the penalty is used to indicate the degree of violation of engineering constraints, or, in other words, the feasibility of the design. The more constraints are violated or the worse a single constraint is violated, the higher the penalty is and the less feasible the design becomes. Here, the idea is illustrated by using three penalty functions based on an engine intake model, as examples of representations of explicit rules.

At the core of our knowledge-based repair system lies a surrogate model of this ensemble of penalty functions - here we use a support vector regression (SVR) model. We discuss this next.

B. Support vector regression

Support Vector Regression(SVR) is a form of surrogate modelling that allows predictions to be inferred from pre-computed sets of results — the so called training data. The model produced by SVR only depends on a subset of the training data, because the cost function for building the model ignores any training data that are close (within a threshold ϵ) to the model prediction.

Suppose we have obtained the training data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^k, \forall i \in \{1, \ldots, n\}$ and the outputs from the analysis or, in this case, penalty functions in a vector $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$. A feature of support vector regression is that we are allowed to specify a margin ϵ within which we can tolerate prediction errors. We only accept any error that is less than ϵ. Thus, the prediction function $\hat{f}(x)$ will have at most ϵ deviation from the actually obtained function values for all the training data. On the other hand, it is desired to minimize the model complexity. For example, consider the first order linear regression model

$$\hat{f}(x) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + \mu \tag{1}$$

where $\mathbf{w}$ is the coefficient vector $\{w_0, w_1, \ldots, w_k\}$ and μ is the bias. Its model complexity can be measured by the norm of $\mathbf{w}$, i.e. $\|\mathbf{w}\|^2 = \mathbf{w}^{\mathbf{T}}\mathbf{w}$.

The problem described above can be cast in the form of a convex optimization problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} & \begin{cases} y_i - \mathbf{w}^{\mathbf{T}}\mathbf{x}_i - \mu \le \epsilon \\ \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + \mu - y_i \le \epsilon. \end{cases} \end{array} \tag{2}$$

Notice that the above optimization problem may not be feasible when such a function f, which approximates all pairs $(\mathbf{x}_i, y_i)$ within ϵ precision, does not exist. To tackle this, slack variable pairs ξ^+, ξ^- are introduced to relax the constraints in the original problem in case (2) becomes infeasible. Of course we want these slack variables to be as small as possible. Thus, (2) can be transformed into:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{k}(\xi_i^+ + \xi_i^-) \\ \text{subject to} & \begin{cases} y_i - \mathbf{w}^{\mathbf{T}}\mathbf{x}_i - \mu \le \epsilon + \xi_i^+ \\ \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + \mu - y_i \le \epsilon + \xi_i^- \\ \xi_i^+, \xi_i^- \ge 0 \end{cases} \end{array} \tag{3}$$

The constant $C \ge 0$ in (3) is a user-defined parameter, which determines the trade-off between model complexity and the amount up to which excessive error can be tolerated. When C approaches 0, the last term in the optimization objective, $C\sum_{i=1}^{k}(\xi_i^+ + \xi_i^-)$, also approaches 0 since the summation of slack variable pairs is always finite. Thus the optimization focuses on minimizing $\frac{1}{2}\|\mathbf{w}\|^2$, and results in a flat prediction. On the other hand, a larger constant C will lead to a closer fitting of the training data, since more emphasis is put on minimizing $\sum_{i=1}^{k}(\xi_i^+ + \xi_i^-)$. So C can be regarded as a cost index: the higher C is, the more costly prediction error becomes.

1. *Finding* $\mathbf{w}$

Having illustrated the basic idea of support vector regression, we pursue a practical solution to the above constrained optimization problem. The key idea is to introduce two sets of Lagrange multipliers:

$$\begin{cases} \eta_i^+ & \ge & 0 \\ \eta_i^- & \ge & 0 \end{cases} \quad \text{and} \quad \begin{cases} \alpha_i^+ & \ge & 0 \\ \alpha_i^- & \ge & 0 \end{cases}$$

which correspond to the constraints

$$\begin{cases} y_i - \mathbf{w}^{\mathbf{T}}\mathbf{x}_i - \mu & \le & \epsilon + \xi_i^+ \\ \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + \mu - y_i & \le & \epsilon + \xi_i^- \end{cases} \quad \text{and} \quad \begin{cases} \xi_i^+ & \ge & 0 \\ \xi_i^- & \ge & 0 \end{cases}$$

in (3) respectively. The Lagrange function is then given by the original objective function minus the sum of all products between Lagrange multipliers and corresponding constraints. Through the process of support vector expansion, it can be proved that:

$$\begin{aligned}\hat{f}(x) &= \sum_{i=1}^{k}\left[(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i\right]\cdot\mathbf{x} + \mu \\ &= \sum_{i=1}^{k}(\alpha_i^+ - \alpha_i^-)(\mathbf{x}_i\cdot\mathbf{x}) + \mu. \end{aligned} \tag{4}$$

In Equation 4, **w** is completely described as a linear combination of the training data. Thus, it is not necessary to compute **w** explicitly to get the prediction model. The $\alpha^{\pm}$ are computed by quadratic programming.

2. Finding μ

After **w** has been found by the support vector expansion, we need to find the bias term μ to complete Eq.(1). The Karush–Kuhn–Tucker conditions is used to find μ. It can be proved that:

$$\mu = y_i - \mathbf{w}^{\mathbf{T}}\mathbf{x}_i - \epsilon \quad \text{if} \quad 0 < \alpha_i^+ < C. \tag{5}$$

In a similar way,

$$\mu = y_i - \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + \epsilon \quad \text{if} \quad 0 < \alpha_i^- < C. \tag{6}$$

3. Kernels

To capture more complicated landscapes of the unknown function, it is also desirable to make the SVR algorithm nonlinear. To achieve this, the kernel functions k, which correspond to a dot product in Equation 4 are widely used. Some widely used kernels are:

- Homogeneous polynomial kernels

$$k(\mathbf{x}_1,\mathbf{x}_2) = (\mathbf{x}_1\cdot\mathbf{x}_2)^p \qquad p \in \mathbb{N} \tag{7}$$

- Inhomogeneous polynomial kernels

$$k(\mathbf{x}_1,\mathbf{x}_2) = (\mathbf{x}_1\cdot\mathbf{x}_2 + c)^p \qquad p \in \mathbb{N}, \quad c > 0 \tag{8}$$

- Gaussian

$$k(\mathbf{x}_1,\mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}\right) \tag{9}$$

- Kriging

$$k(\mathbf{x}_1,\mathbf{x}_2) = \exp\left(-\sum_{i=1}^{k}\theta_k\left|\mathbf{x}_1^{(i)} - \mathbf{x}_2^{(i)}\right|^{p_k}\right) \tag{10}$$

Now we can restate the optimization problem with the kernel concept included:
Maximize:

$$\begin{aligned}&\sum_{i=1}^{k} y_i(\alpha_i^+ - \alpha_i^-) - \epsilon\sum_{i=1}^{k}(\alpha_i^+ + \alpha_i^-) \\ &\qquad - \frac{1}{2}\sum_{i,j=1}^{k}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)k(\mathbf{x}_i,\mathbf{x}_j)\end{aligned} \tag{11}$$

subject to:

$$\sum_{i=1}^{k}(\alpha_i^+ - \alpha_i^-) = 0 \quad \text{and} \quad \alpha_i^{\pm} \in [0, C].$$

4. Computing ϵ using ν-SVR

In many cases we are able to estimate ϵ based on our understanding of the engineering problem at hand. However, when such information is unavailable, we can still calculate a value of ϵ via the ν-SVR method.

In ν-SVR, the constrained optimization problem can be represented as:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\nu\epsilon + \sum_{i=1}^{k}(\xi_i^+ + \xi_i^-)\right) \\ \text{subject to} & \begin{cases} y_i - \mathbf{w^T}\mathbf{x}_i - \mu \leq \epsilon + \xi_i^+ \\ \mathbf{w^T}\mathbf{x}_i + \mu - y_i \leq \epsilon + \xi_i^- \\ \xi_i^{\pm} \geq 0, \epsilon \geq 0. \end{cases} \end{array} \tag{12}$$

Following a similar procedure, (12) can be transformed into the dual optimization problem:
Maximize:

$$\sum_{i=1}^{k} y_i(\alpha_i^+ - \alpha_i^-) \tag{13}$$
$$- \frac{1}{2}\sum_{i,j=1}^{k}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to:

$$\sum_{i=1}^{k}(\alpha_i^+ - \alpha_i^-) = 0, \quad \alpha_i^{\pm} \in [0, C]$$

and

$$\sum_{i=1}^{k}(\alpha_i^+ + \alpha_i^-) \leq C\nu.$$

To get the value of ϵ, we equate (5) and (6):

$$\begin{aligned} \epsilon \;=\; & \frac{1}{2}\Bigg(y_m - y_n - \sum_{i=1}^{k}(\alpha_i^+ - \alpha_i^-)k(\mathbf{x}_i, \mathbf{x}_m) \\ & + \sum_{i=1}^{k}(\alpha_i^+ - \alpha_i^-)k(\mathbf{x}_i, \mathbf{x}_n)\Bigg) \end{aligned} \tag{14}$$

if

$$0 < \alpha_m^+ < C \quad \text{and} \quad 0 < \alpha_n^- < C.$$

C. Geometry repair

The SVR surrogate can be used to predict geometry quality after it is properly trained with a sufficient amount of data representing penalty function values. Then, when a flawed geometry is detected, an alternative set of design variables can be found automatically using an optimization-based search over the surrogate, making an automatic geometry repair process possible.

The repair of a low quality geometry can be implemented by identifying the design alternative with the smallest possible repair alteration (SPRA). To be precise, the SPRA is the set of design variable increments that will make the design feasible while keeping changes to a minimum, thus retaining the original design intent to the maximum. The method will be further elaborated in Section III.

A flowchart of the proposed process is shown in Figure 1. In the preparation phase, expert knowledge is translated into several engineering rules, which are in turn translated into penalty functions. At the same time, a sampling plan is generated across the design space. The sampled geometries are evaluated with penalty functions supplied by the designer. In the second step, evaluations and sampling plans are stored in a database, which is used to generate a surrogate model. This is tuned before it is used to predict geometry quality and search for the nearest feasible geometries if the new unknown geometry is predicted to

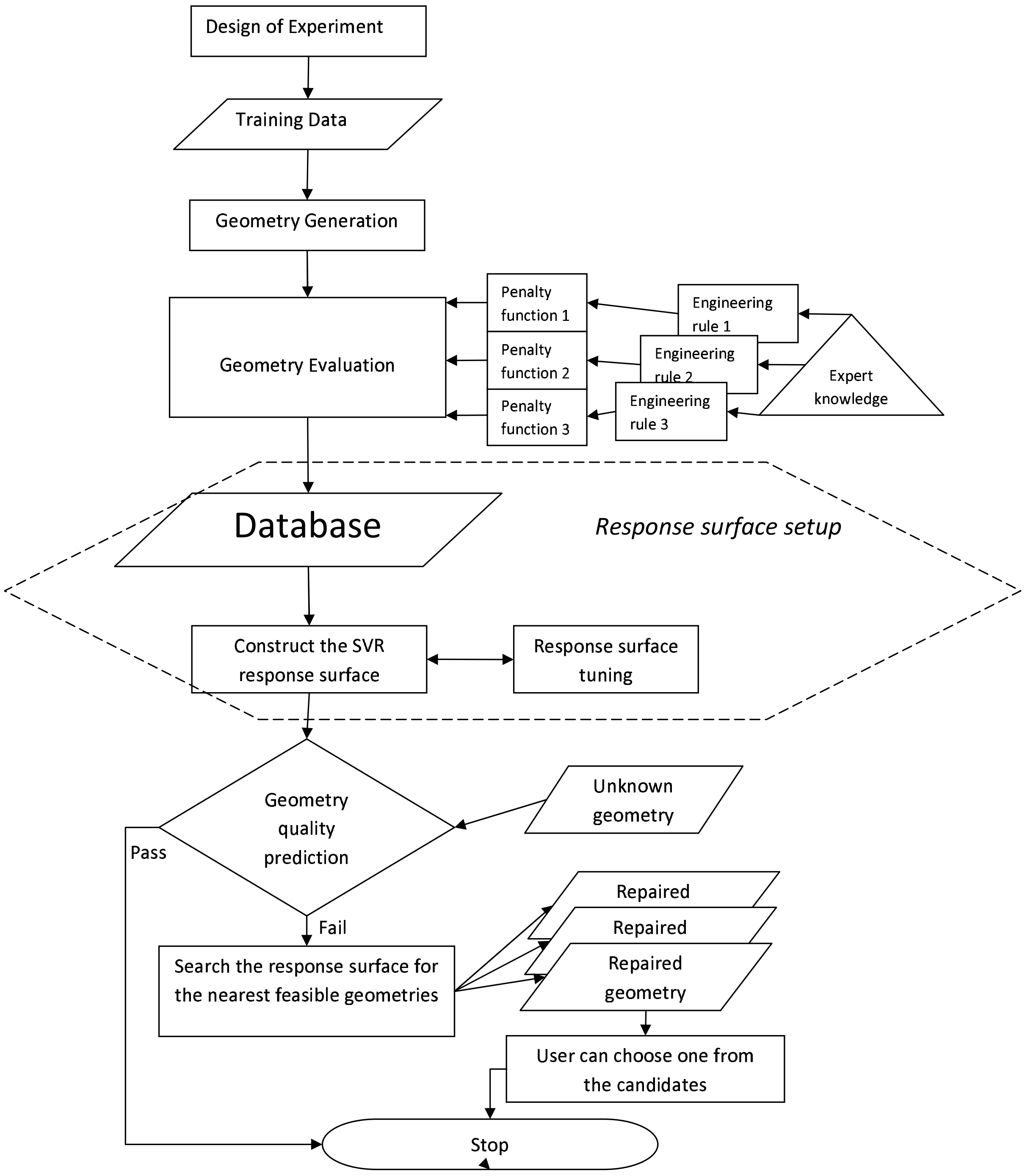


Figure 1. Flowchart: Repair process

be infeasible. The algorithm used here generates a set of repaired geometries, from which a user can choose one.

The choice of method to be employed for the search for a repair alternative is fairly open from the standpoint of computing budget because both the objective (the distance between the original design and repair alternatives) and the constraint (the SVR geometry quality predictor) is cheap to evaluate. However, if a simple gradient-based local search is employed, it is possible that the true nearest possible repair alternative is neglected if it is 'concealed' by a local 'hump' of the surrogate. In our work, a variable resolution evolutionary operation (EVOP) approach has been adapted to avoid such problems.[4] The optimization process starts with an initial global search, which covers the design space. If the initial global search is successful, further local searches are repetitively performed within the hyper-sphere which centred on the original design point with a radius equal to the distance between nearest feasible design found so far. In each repetition, or "generation" in EVOP terminology, a series of offspring are obtained by using the full factorial sampling technique. If a whole generation fails to produce a better design alternative, the sampling density will be increased to allow a more intense search. After a successful round of search, the design variable sets that fulfil the quality constraint are listed, the one that is closest to the original design picked and used as the benchmark for the next round of search. The search terminates when either the optimized design reaches a pre-determined penalty value below which the design is deemed as "satisfactory" or the limit of computing budget is reached.

III. Case Study: A two-dimensional Aeroengine Intake Design and Repair

In this section, we use a simple 2-dimensional parameterized inlet model as a test case, as shown in Figure 2. The external shape of the airframe, the position of the engine and rear bulkhead are fixed. The

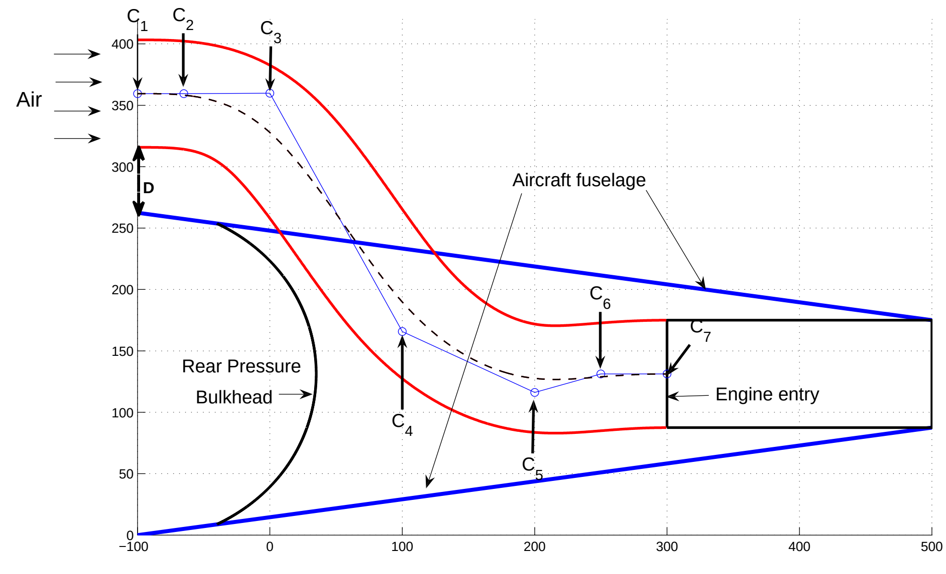


Figure 2. A simplified 2D intake model

intake shape is entirely dependent upon its center axis (the dashed curve in the figure), which is a B-spline defined by seven control points (C1–C7, as noted in the figure). The horizontal positions of these control points are held constant. The vertical positions of C1 and C2 are kept the same so that the entrance of the intake stays horizontal. Additionally, the vertical position of C6 and C7 are on the engine centreline to ensure that the exit of the intake is level and connects smoothly with the engine. The cross-sectional area of the intake duct equals that of the engine face and is held constant along the duct center axis. The vertical positions of other points are left free as design variables. Horizontal positions, acceptable vertical position ranges and corresponding design variables for each control point are listed in Table 1. The design variable set $\mathbf{x}$ comprises four design variables in our test case:

$$\mathbf{x} = \{x_1, x_2, x_3, x_4\}.$$

For example, the design variable set $\mathbf{x} = \{359.5, 359.9, 165.8, 116.1\}$ will result in the design shown in Figure 2.

Control Point	Horizontal position	Vertical range	Design variable
C_1	-100	245 – 385	x_1
C_2	-65	same as C_1	same as C_1
C_3	0	175 – 455	x_2
C_4	100	70 – 350	x_3
C_5	200	35 – 315	x_4
C_6	250	131.25	
C_7	300	131.25	

Table 1. List of the horizontal positions, acceptable vertical position ranges and corresponding design variables for each control point

A. Knowledge representation

As mentioned in section A, engineering knowledge in the form of explicit rules can be transformed into penalty functions in order to make them optimizable objectives. In this section, the idea is illustrated by three examples. First of all, an intake position penalty function P_1 is related to the vertical distance D between the aircraft fuselage (upper boundary) and the intake lower boundary at the air intake entrance position, as show in Figure 2. A positive D corresponds to a protruding intake design, which will increase the aerodynamic drag. On the other hand, negative distance will result in an intake design where the entrance is partially submerged into the fuselage. Since the air capture area is reduced, the capture/throat area ratio is reduced, which is undesirable from an intake aerodynamics point of view. Such engineering considerations are incorporated into P_1, in which both unfavorable scenarios receive a penalty. We consider P_1 as having the form

$$P_1 = \begin{cases} D^{3/2} & \text{if } D \geq 0 \\ -100D & \text{if } D < 0 \end{cases}. \tag{15}$$

The reason we choose an exponential function when $D \geq 0$ and a linear function when $D < 0$ here is because we would like to take different forms of penalty function in order to test the learning ability of support vector regression. Since the airframe external shape is fixed and the vertical position of the intake entrance is defined by design variable x_1, D is solely dependent upon x_1. Here x_1 and D are related by

$$D = x_1 - 306.25, \tag{16}$$

so that P_1 can be rewritten as a function of the design variables

$$P_1 = P_1(\mathbf{x}) = \begin{cases} (x_1 - 306.25)^{3/2} & \text{if } x_1 - 306.25 \geq 0 \\ -100(x_1 - 306.25) & \text{if } x_1 - 306.25 < 0. \end{cases} \tag{17}$$

A second penalty function P_2 is related to the curvature of the intake duct. It can be seen that certain design variable combinations can render the overall shape of the intake duct convoluted. From an aerodynamic engineer's point of view, designs with sharp bends are unfavorable because at the sharp bends, air flow can separate and cause distorted pressure fields on the engine face. Furthermore, too high a curvature of the center line can cause a loop in its upper and lower offset curves and render the design impractical. Such a failed design is shown in Figure 3, in which the design variables are set to be $\mathbf{x} = \{360, 300, 50, 250\}$. Therefore, to represent such engineering concerns, a penalty function P_2 is set up to penalize geometries with

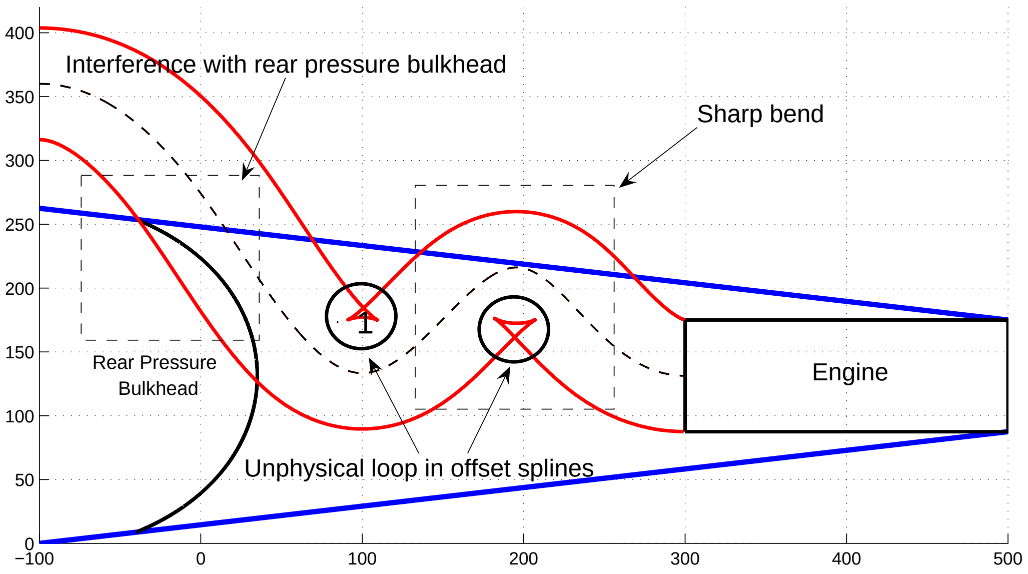


Figure 3. Failed design example due to high curvature

excessive curvature. The curvature of the the center B-spline axis can be computed via finding its second order derivative information.[5] The curvature at u, $k(u)$, can be computed as follows:

$$k(u) = |\mathbf{f}'(u) \times \mathbf{f}''(u)|/|\mathbf{f}'(u)|^3 \quad . \tag{18}$$

The radius of curvature $R(u)$ is the reciprocal of curvature $k(u)$. A loop will occur in the offset curve if $R(u)$ is less than the radius of the engine face, which is 43.75 in this intake design case. P_2 is set up as a sum of $R(u)$s whenever its value is less than 43.75. i.e.

$$P_2 = \sum_{u=0}^{1} R(u), \forall R(u) < 43.75 \quad . \tag{19}$$

To set up a third penalty function P_3, it is noted that those duct designs which interfere with the rear pressure bulkhead are undesirable since precious space in the fuselage is occupied and the bulkhead may become structurally inefficient. Such an unfavorable geometry is illustrated in Figure 4. A penalty function P_3 is set up to penalize interference between the two parts. A more severe interference will incur a higher penalty value of P_3.

So far, we have set up three penalty functions for three explicit rules, each of which represents some form of engineering knowledge. Compared to direct consultation with a human expert or evaluation of CFD codes, explicit rules are relatively cheap to calculate and thus more favorable in the process of generating training data. However, in contrast to the four design variables in our test case, many more design variables may exist in real engineering applications with underlying interactions that the designer may be unaware of. In the region of the search space where these explicit rules cannot reach, we can set up a CFD analysis and/or consult human experts to get information about the deficiencies of the geometry. After all, if we had an exhaustive analytical rule base that covered the whole design space, we would not need a regression model at all. Human expert consultation and CFD analysis will result in case-based knowledge, which can be used in the training of a surrogate.[3]

B. Repair result

Following the procedure that has been described in the previous sections, it is possible to predict the quality of the geometry and find an SPRA for faulty geometries. To test the idea of geometry repair, we begin

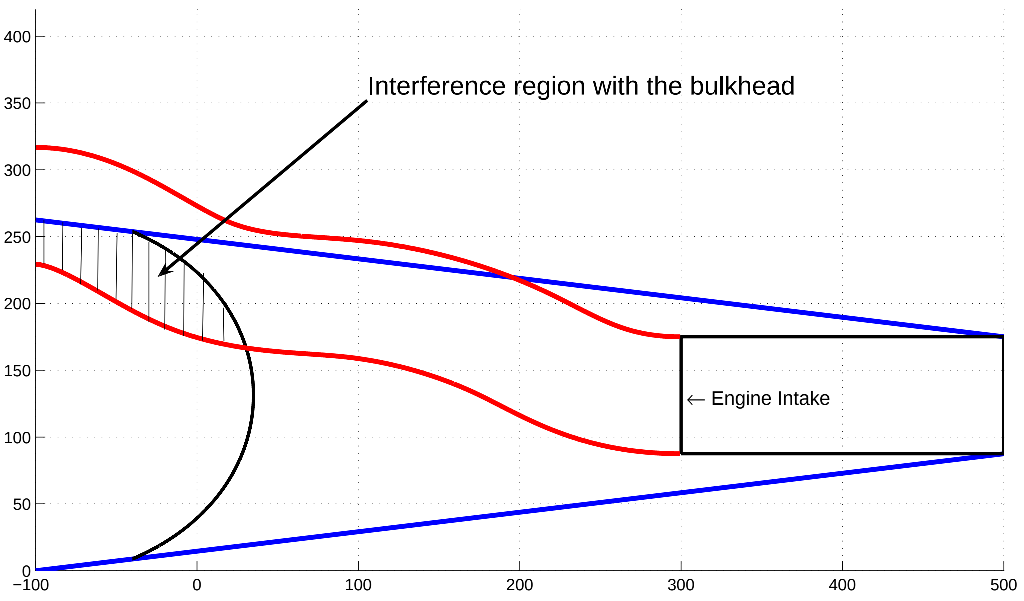


Figure 4. A faulty geometry due to structure interference

with the single penalty function P_3 which penalizes the interference between the rear pressure bulkhead and intake, and a pre-determined penalty threshold value $p_{th} = 1500$. A faulty design whose corresponding design variable set is $\mathbf{x} = [0.2, 0.1, 0.5, 0.5]$ is shown in Figure 4. It is obvious that the interference between the bulkhead and intake is severe.

To begin with, 100 training points $\mathbf{x}_i, i = 1, 2, \ldots, 100$ are chosen via the Latin Hypercube sampling method. For each $\mathbf{x}_i$, $y_i = P_3(\mathbf{x}_i)$. The SVR geometry quality predictor $\hat{f}(\mathbf{x})$ is trained on these $(\mathbf{x}_i, y_i)$ pairs,with $i = 1, 2, \ldots, 100$. The predicted penalty for this design is found to be $\hat{f}(\mathbf{x}) = 2.7874 \times 10^4$. Following a search started globally over the whole design region, a Pareto front was determined (the circled points in Figure 5). The point which lay below and is closest to the penalty threshold line (black dashed line) was chosen as the current best point in the search process (black solid point in Figure 5). It was indicated by the SVR predictor that the chosen point had a penalty of 940 and its Euclidean distance from the original design was 0.4743.

After the successful initial global search (with a sample size of 11^4), EVOP continues by repetitively performing local searches within the hyper-sphere centered at the original design point with a radius equal to the distance from the best point found so far. The sampling points outside the hyper-sphere are discarded before they get evaluated to save computing budget. This strategy proves to be able to improve the search result in each of the four consecutive rounds before it converged at $\mathbf{x} = [0.2650, 0.4900, 0.6950, 0.5650]$ with $\hat{f}(\mathbf{x}) = 1440$. The geometry corresponding to optimized $\mathbf{x}$ is shown in Figure 6. Comparing Figure 4 and Figure 6, it can been seen that in the optimized design the interference between the rear pressure bulkhead and the engine intake had been significantly reduced, while the aft part of the intake was subject only to minor changes. This indicates that the design can be made feasible by the repair process while the original design intent is broadly retained.

Next we investigate the application of the method on a design candidate with multiple flaws. Such a design is presented in Figure 7 with

$$\mathbf{x}_s = [0.45, 0.35, 0.05, 0.65].$$

The intake entrance of the design is slightly submerged in the fuselage and the duct itself is snaky and interferes with the bulkhead.

In this case, three penalty functions are combined to form a single penalty function $P = P_1 + P_2 +$

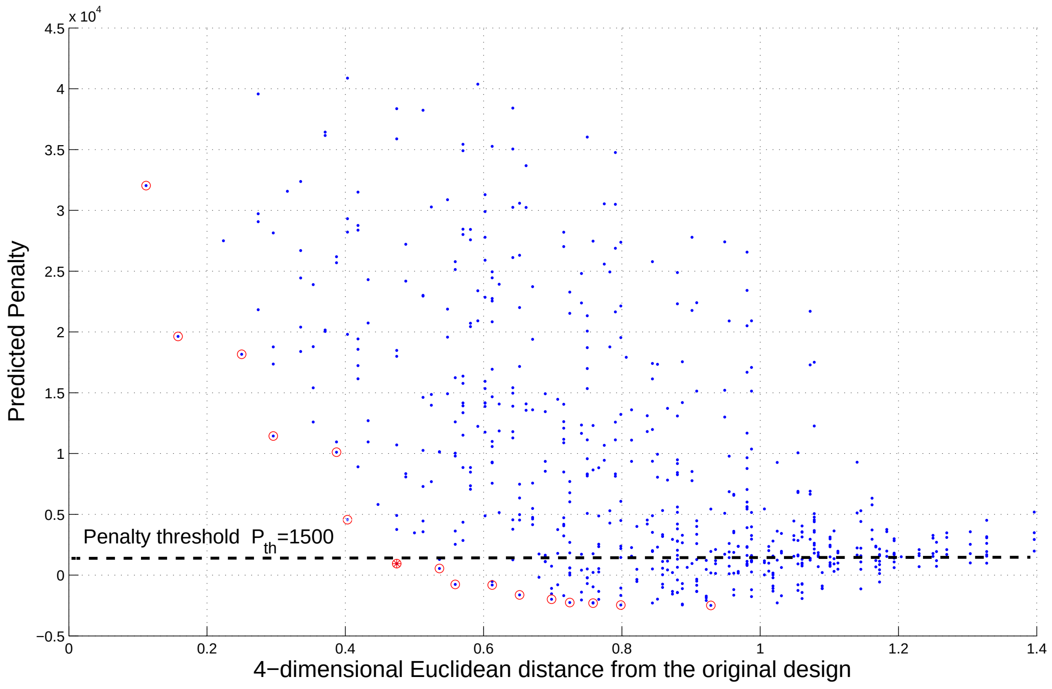


Figure 5. Initial global search result with its Pareto front

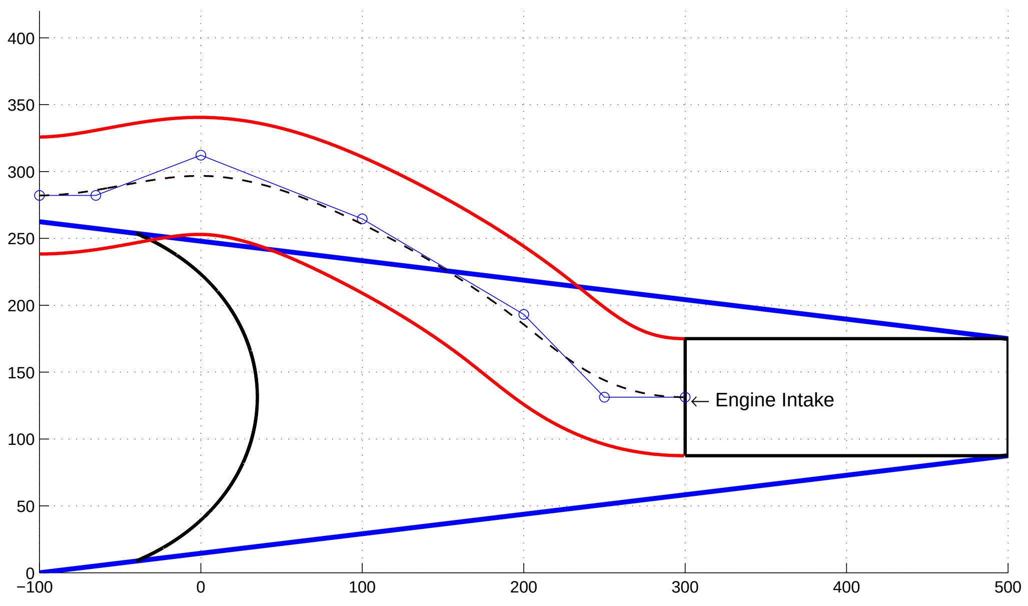


Figure 6. Suggested repair alternative of the original design

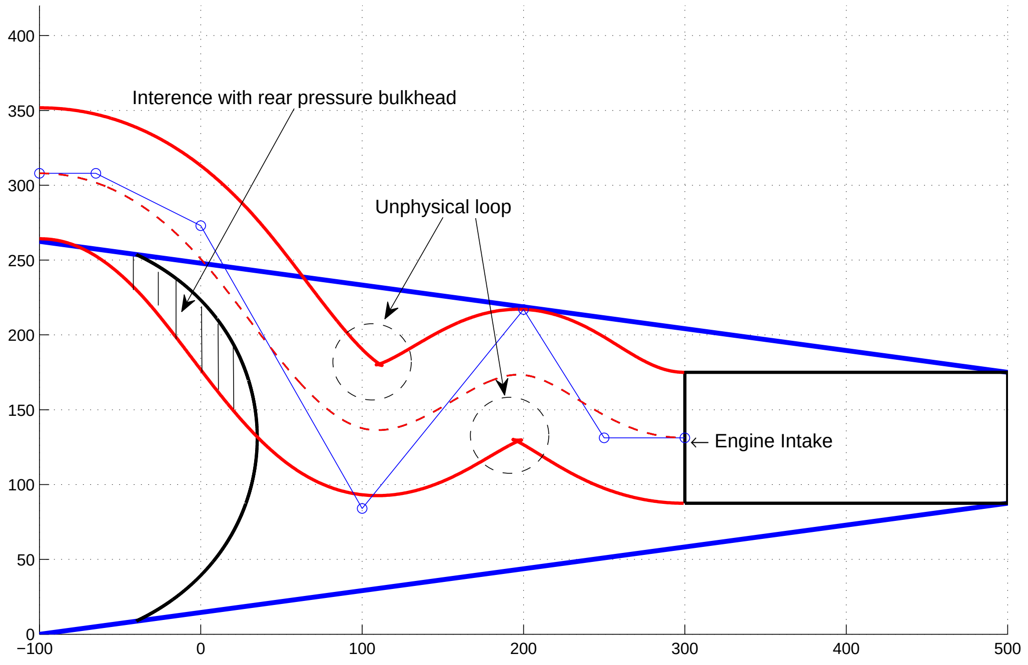


Figure 7. A design candidate with multiple flaws

P_3. The same set of training points $\mathbf{x}_i$ are used but this time $y_i = P(\mathbf{x}_i)$. An SVR surrogate $\hat{f}(\mathbf{x})$ is trained on the data and is used for the optimization. In this case, we use 10 different p_{th} values: $p_{th} = 0, \hat{f}(\mathbf{x}_s)/10, 2\hat{f}(\mathbf{x}_s)/10, \ldots, 9\hat{f}(\mathbf{x}_s)/10$ which lead to 10 different repair suggestions and leave the final option to the engineer. The repair alternatives are presented in Table 2, (notice that the predicted penalty for the original design $\mathbf{x}_o$ is $\hat{f}(\mathbf{x}_s) = 5642$):

Table 2: Design alternatives with different penalty threshold levels

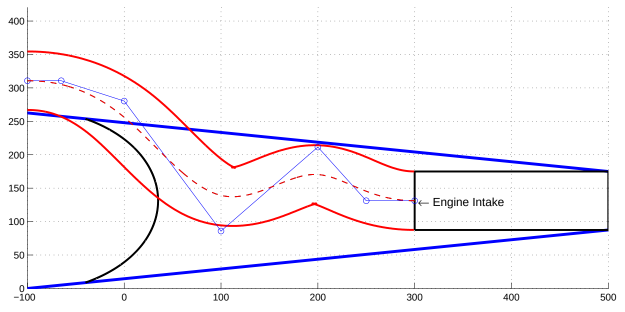
p_{th}	$\mathbf{x}$	Repair alternative suggestion
$9\hat{f}(\mathbf{x}_s)/10$ $= 5053$	$[0.4696, 0.3761, 0.0565, 0.6304]$	

Table 2: (continued)

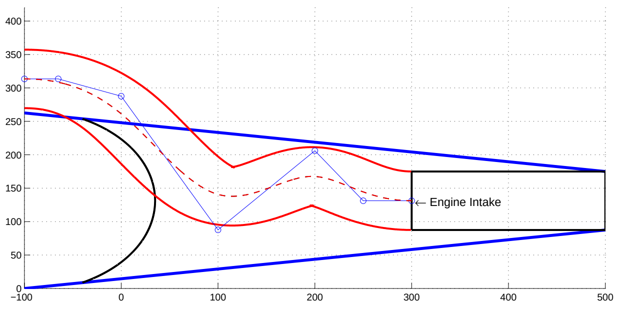
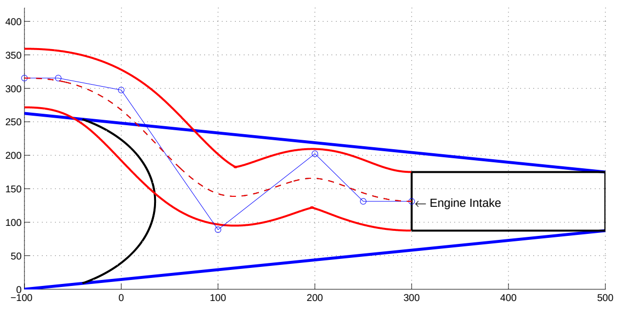
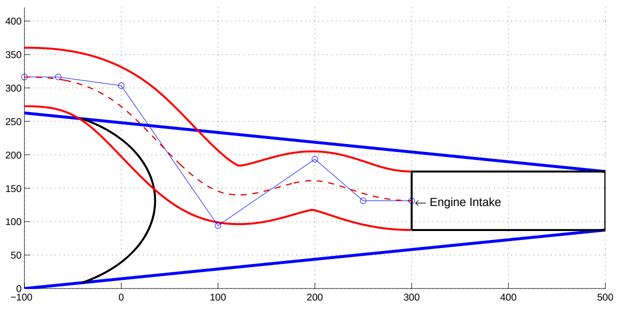
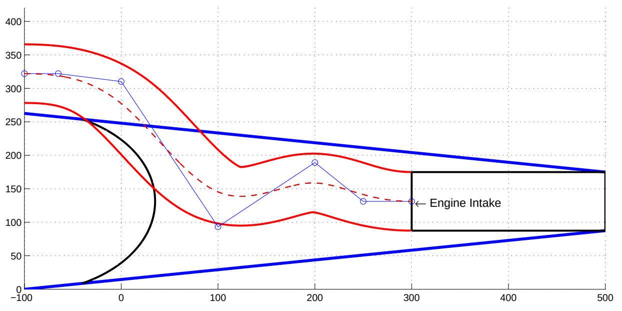
p_{th}	$\mathbf{x}$	Repair alternative suggestion
$8\hat{f}(\mathbf{x}_s)/10$ $= 4492$	$[0.4892, 0.4022, 0.0631, 0.6108]$	
$7\hat{f}(\mathbf{x}_s)/10$ $= 3930$	$[0.5025, 0.4375, 0.0675, 0.5975]$	
$6\hat{f}(\mathbf{x}_s)/10$ $= 3369$	$[0.5103, 0.4585, 0.0862, 0.5656]$	
$5\hat{f}(\mathbf{x}_s)/10$ $= 2807$	$[0.5496, 0.4828, 0.0832, 0.5504]$	

Table 2: (continued)

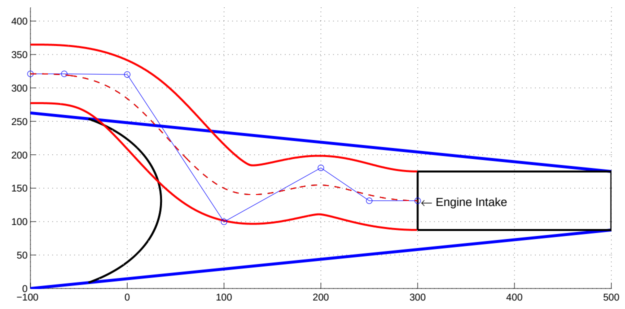
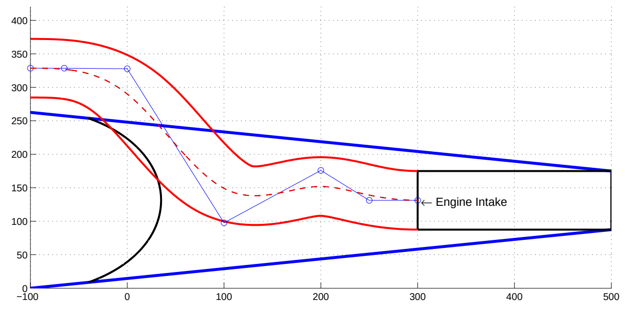
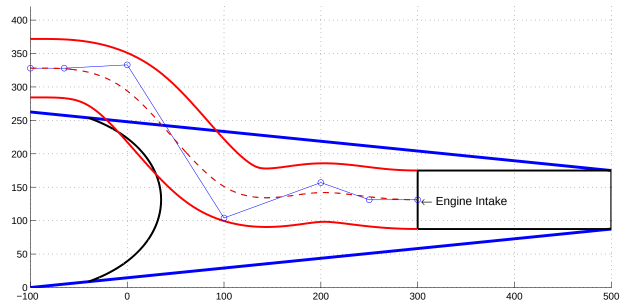
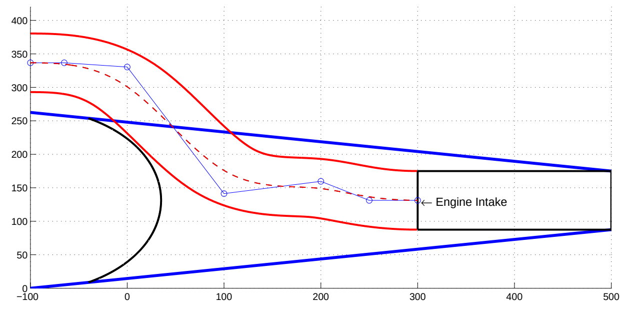
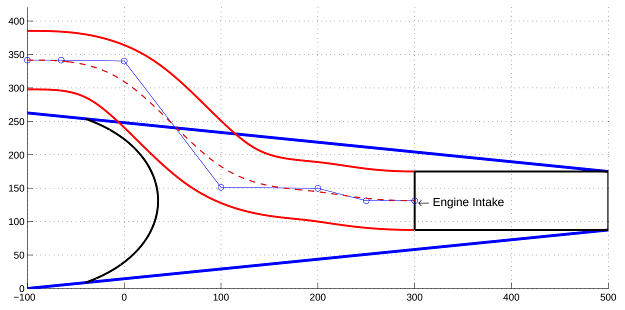
p_{th}	$\mathbf{x}$	Repair alternative suggestion
$4\hat{f}(\mathbf{x}_s)/10 = 2246$	[0.5434,0.5181,0.1060,0.5193]	
$3\hat{f}(\mathbf{x}_s)/10 = 1684$	[0.5966,0.5454,0.0989,0.5034]	
$2\hat{f}(\mathbf{x}_s)/10 = 1123$	[0.5929,0.5644,0.1215,0.4356]	
$1\hat{f}(\mathbf{x}_s)/10 = 561$	[0.6548,0.5548,0.2548,0.4452]	

Table 2: (continued)

p_{th}	$\mathbf{x}$	Repair alternative suggestion
0	[0.6900,0.5900,0.2900,0.4100]	

It can be seen that the repair suggestions $\mathbf{x}_r$ gradually change from being very similar to the original design to a "near perfect design" as the penalty threshold p_{th} gradually decreases. The part exhibiting high curvature is gradually smoothed, while at the same time the interference with fuselage and rear bulkhead are gradually reduced. The engineer can thus choose one of these optimized design alternatives in a way that best balances between the desire to retain the original design intent and the 'health' of the geometry.

C. Illustration of the Repair path

To better illustrate the repair path on the predicted penalty function landscape, the process is slightly modified so that two of the design variables are fixed, allowing the other two variables to vary and to be optimized. The optimization problem is thus reduced to two dimensions. The original design labeled as No.1 in Figure 8 is $\mathbf{x} = [0.4, 0.2, 0.2, 0.9]$. In this case, the second and fourth design variable (x_2 and x_4) are left as variables. The contour plot on the left of Figure 8 shows the predicted penalty function landscape with regard to x_2 and x_4, along with the repair path and changing geometries.

IV. Conclusion

In modern multi-objective design optimization an effective geometry engine is becoming an essential tool and its performance has a significant impact on the entire MDO process. Building a parametric geometry requires difficult compromises between the conflicting goals of robustness and flexibility. The work presented here provides a solution for improving the robustness of a parametric geometry by capturing and modelling engineering knowledge, and deploying it automatically in the search for a more robust design alternative, while supporting the original design intent. Although the 2D intake design repair example used seems a simple problem and the repair direction can be predicted by an observant reader, the proposed method has the potential to greatly reduce the engineer's workload and increase the design efficiency and robustness when it is applied on real scale engineering design problems.

Acknowledgments

The authors would like to thank Rolls-Royce Group plc, the Engineering and Physical Sciences Research Council (EPSRC) and the Royal Academy of Engineering for the financial support of this work.

References

[1]Roy, R., Hinduja, S., and Teti, R., "Recent advances in engineering design optimisation: Challenges and future trends," *CIRP Annals - Manufacturing Technology*, Vol. 57, No. 2, 2008, pp. 697–715.

[2]Samareh, J., "Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization," *AIAA Journal*, Vol. 39, No. 5, May 2001, pp. 877–884.

[3]Sóbester, A. and Keane, A., "Supervised learning approach to parametric computer-aided design geometry repair," *AIAA Journal*, Vol. 44, No. 2, 2006, pp. 282 – 289.

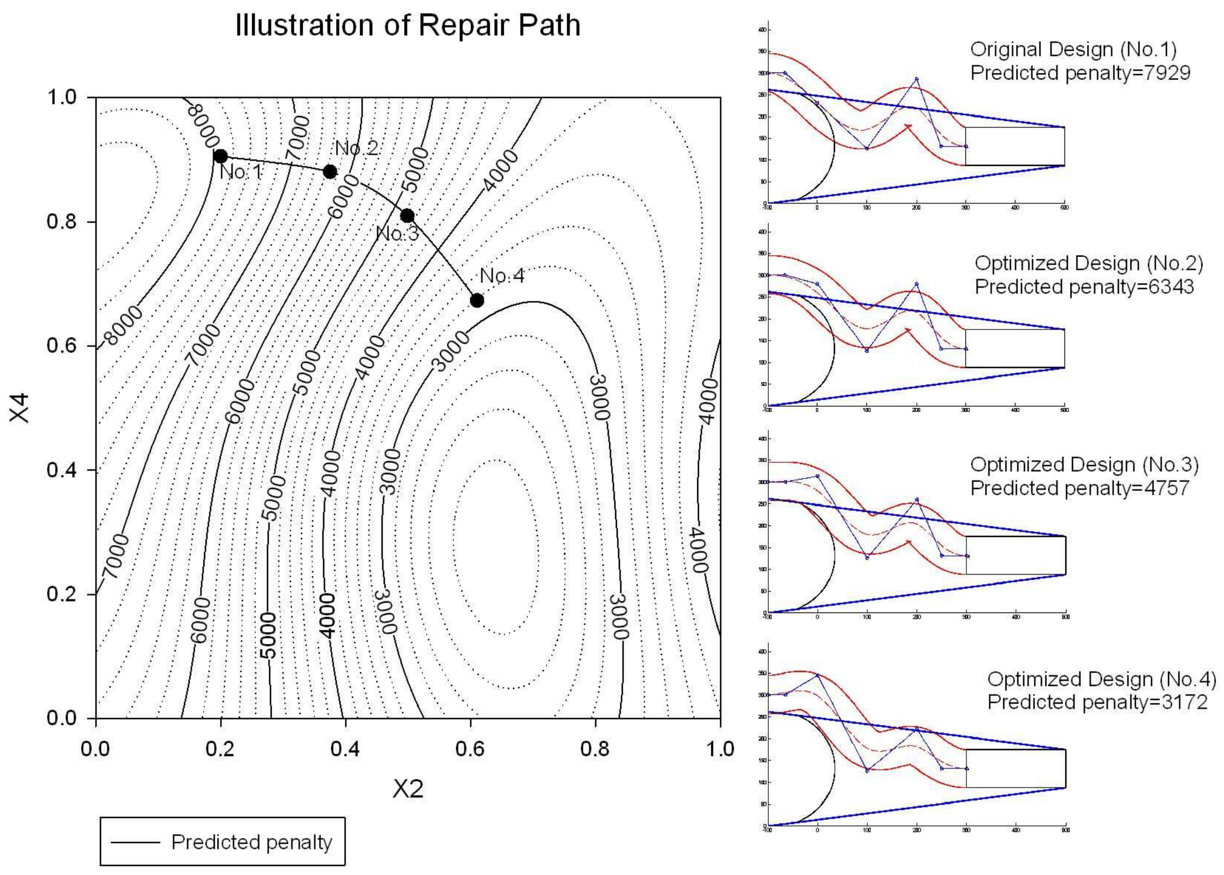


Figure 8. Illustration of a Repair Path

[4]Beyer, H. and Schwefel, H., “Evolution strategies–A comprehensive introduction,” *Natural Computing*, Vol. 1, No. 1, 2002, pp. 3–52.

[5]Piegl, L. and Tiller, W., *The NURBS book*, Monographs in visual communications, Springer, 2nd ed., 1997.