RBF-based mesh morphing improvement using Schur complement applied to rib shape optimization

Franck Mastrippolito
Univ. Lyon, Ecole Centrale de Lyon, Laboratoire de Mécanique des Fluides et d'Acoustique UMR5509, Ecully, France and Univ. Grenoble Alpes, CEA, LITEN, DTBH, Laboratoire Echangeurs et Réacteurs, Grenoble, France

Stephane Aubert
Univ. Lyon, Ecole Centrale de Lyon, Laboratoire de Mécanique des Fluides et d'Acoustique UMR5509, Ecully, France

Frédéric Ducros
Univ. Grenoble Alpes, CEA, LITEN, DTBH, Laboratoire Echangeurs et Réacteurs, Grenoble, France, and

Martin Buisson
Univ. Lyon, Ecole Centrale de Lyon, Laboratoire de Mécanique des Fluides et d'Acoustique UMR5509, Ecully, France

Received 18 June 2018
Revised 19 November 2018
20 December 2018
Accepted 20 December 2018

Abstract

Purpose – This paper aims to improve the radial basis fuction mesh morphing method. During a shape optimization based on computational fluid dynamic (CFD) solvers, the mesh has to be changed. Two possible strategies are re-meshing or morphing. The morphing one is advantageous because it preserves the mesh connectivity, but it must be constrained.

Design/methodology/approach – RBF mesh deformation is one of the most robust and accurate morphing method. Using a greedy algorithm, the computational cost of the method is reduced. To evaluate the morphing performances, a rib shape optimization is performed using the NSGA-II algorithm coupled to kriging metamodels based on CFD. The morphing method is then compared to a re-meshing strategy.

Findings – The authors propose a method, based on Schur complement, to speed-up the greedy process. By using the information of the previous iteration, smaller linear systems are solved and time is saved. The optimization results highlight the interest of using a morphing-based metamodel regarding the resolution time and the accuracy of the interpolated solutions.

Originality/value – A new method based on Schur complement is addressed to speed-up the greedy algorithm and successfully applied to a shape optimization.

Keywords Optimization, CFD, Radial basis function, Mesh deformation, Schur complement, Metamodels

Paper type Research paper

International Journal of Numerical Methods for Heat & Fluid Flow
Vol. 30 No. 9, 2020
pp. 4241-4257

0961-5539
DOI 10.1108/HFF-06-2018-0309

Nomenclature

$\vec{b}$ = Second unit tangent vector, $\sum_{i=1}^{3} b_i \vec{e}_i$ ($\in \mathbb{R}^3$);

C_f = Head loss coefficient ($\in \mathbb{R}$);

D_H = Hydraulic diameter($\in \mathbb{R}$);

d = Distance measuring function ($\mathbb{R}^3 \rightarrow \mathbb{R}^+$);

D = Parameters space ($\subset \mathbb{R}^3$);

$\vec{e}_i$ = ith basis vector ($\in \mathbb{R}^3$);

e = Rib top width ($\in \mathbb{R}$);

E = Rib base width ($\in \mathbb{R}$);

h_{rib} = Rib height ($\in \mathbb{R}$);

H = Channel height ($\in \mathbb{R}$);

L = Rib pitch ($\in \mathbb{R}$);

$\vec{n}$ = Unit normal vector, $\sum_{i=1}^{3} n_i \vec{e}_i$ ($\in \mathbb{R}^3$);

N = Number of mesh nodes ($\in \mathbb{N}$);

N_a = Number of control points in the Schur subset ($\in \mathbb{N}$);

N_c = Number of control points ($\in \mathbb{N}$);

N_g = Number of control points in the greedy subset ($\in \mathbb{N}$);

N_m = Number of moving points ($\in \mathbb{N}$);

N_s = Number of sliding points ($\in \mathbb{N}$);

Nu = Nusselt number ($\in \mathbb{R}$);

N_{res} = Number of control points before restarting ($\in \mathbb{N}$);

$\vec{p}$ = Vector of parameters ($\in \mathcal{D}$);

$\dot{q}$ = Heat flux density ($\in \mathbb{R}$);

r_0 = Support radius ($\in \mathbb{R}$);

$s_{\max}$ = Largest displacement magnitude ($\in \mathbb{R}$);

$\vec{s}$ = Displacement function ($\mathbb{R}^3 \rightarrow \mathbb{R}^3$);

$\vec{s}_{mi}$ = Displacement vector of i^{th} moving point ($\in \mathbb{R}^3$);

$S_\square$ = Vector ($\in \mathbb{R}^{N_m}$);

S^N = Matrix ($\in \mathbb{R}^{N \times 3}$);

$S^\square$ = Vector ($\in \mathbb{R}^{3N_\square}$);

Sch = Schur complement matrix ($\in \mathbb{R}^{3N_a \times 3N_a}$);

$\vec{t}$ = First unit tangent vector, $\sum_{i=1}^{3} t_i \vec{e}_i$ ($\in \mathbb{R}^3$);

T = Temperature ($\in \mathbb{R}$);

U_{deb} = Bulk velocity ($\in \mathbb{R}$);

$\vec{x}$ = Position vector ($\in \mathbb{R}^3$);

$\vec{x}_{ci}$ = Position vector of ith control point ($\in \mathbb{R}^3$);

$\vec{x}_{mi}$ = Position vector of ith moving point ($\in \mathbb{R}^3$);

$\vec{x}_{si}$ = Position vector of ith sliding point ($\in \mathbb{R}^3$);

X = Matrix ($\in \mathbb{R}^{N \times 3}$);

ΔP = Pressure drop ($\in \mathbb{R}$);

$\vec{\gamma}_i$ = Weight vector of ith control point ($\in \mathbb{R}^3$);

$\Gamma_\square$ = Vector ($\in \mathbb{R}^{N_c}$);

Γ^N = Matrix ($\in \mathbb{R}^{N_c \times 3}$);

$\Gamma^\square$ = Vector ($\in \mathbb{R}^{3N_\square}$);

λ = Thermal conductivity ($\in \mathbb{R}$);

ϕ = Radial basis function ($\mathbb{R}^+ \rightarrow \mathbb{R}$);

Φ = Matrix ($\in \mathbb{R}^{N\times N_c}$);
Φ^c = Matrix ($\in \mathbb{R}^{N_c\times N_c}$);
Φ^m = Matrix ($\in \mathbb{R}^{N_m\times N_c}$);
Φ^s = Matrix ($\in \mathbb{R}^{N_s\times N_c}$);
Φ^t = Matrix ($\in \mathbb{R}^{N_s\times N_c}$);
$\Phi^\bullet$ = Matrix ($\in \mathbb{R}^{3N_g\times 3N_g}$);
Φ° = Matrix ($\in \mathbb{R}^{3N_a\times 3N_a}$);
$\Phi^\triangle$ = Matrix ($\in \mathbb{R}^{3N_g\times 3N_a}$);
$\Phi^\triangledown$ = Matrix ($\in \mathbb{R}^{3N_a\times 3N_g}$);
ρ = Density ($\in \mathbb{R}$);
θ = Rib angle ($\in \mathbb{R}$);
0_s = Null vector ($\in \mathbb{R}^{N_s}$);
$0_{s\times m}$ = Null matrix ($\in \mathbb{R}^{N_s\times N_m}$);
$0_{m\times c}$ = Null matrix ($\in \mathbb{R}^{N_m\times N_c}$);
$I_{s\times s}$ = Identity matrix ($\in \mathbb{R}^{N_s\times N_s}$);
$\square^*$ = After deformation state;
$(\square)'$ = Previous greedy iteration state;

$\overrightarrow{u}\cdot\overrightarrow{v}$ = Dot product, $\sum_{i=1}^{3} u_i v_i$ $\left((\mathbb{R}^3,\mathbb{R}^3)\rightarrow\mathbb{R}\right)$; and

$||\overrightarrow{u}||$ = Euclidean norm, $\sqrt{\overrightarrow{u}\cdot\overrightarrow{u}}$ $(\mathbb{R}^3\rightarrow\mathbb{R})$.

1. Introduction

Rib-roughened channels are a common way to achieve efficient blade cooling in gas turbine engines. Several numerical and experimental studies have shown that the rib shape has a strong influence on the aero-thermal behavior (Kim *et al.*, 2010; Rau *et al.*, 1998; Liu *et al.*, 2017). A rib shape optimization based on a genetic algorithm (GA) is proposed in the present work. The bi-objective problem is classically to maximize the heat transfer, expressed through the Nusselt number, while minimizing the head loss coefficient (Luo *et al.*, 2018). These performances are obtained using computational fluid dynamics (CFD) simulations of a periodic rib. To solve the problem, the GA needs numerous evaluations of the performances. As the computational cost of one evaluation is large, a metamodel is used (Simpson *et al.*, 2001).

To build the metamodel, several rib shapes are computed. There are two ways to propagate the shape changes to the mesh, namely, by using CAD-based geometry and re-meshing the domain or by moving the mesh boundaries (Mohammadi and Pironneau, 2004). The second strategy, known as mesh morphing, presents several advantages:

- a robust automatic meshing procedure is not mandatory and the meshing technical know-how may be preserved; and
- the mesh connectivity is preserved, so calculation restarts are possible, without requiring data interpolation and these restarts may speed up the resolution.

On the other hand, the entire parameters space may not be attainable using morphing because the mesh displacement is highly strained by grid-quality constraints (skewness, aspect ratio, volume positivity), boundary conditions (wall, periodic, symmetric) or prescribed surface deformations.

Various techniques for mesh morphing have been proposed, including a spring analogy (Batina, 1990), a pseudo-solid approach (Nielsen and Anderson, 2002) and free form deformation (Sederberg and Parry, 1986). However, to propagate displacements known only

at clouds of control points to the whole volume mesh, the radial basis function (RBF) interpolation method is more and more popular (Jakobsson and Amoignon, 2007). This free mesh connectivity method is flexible enough to deal with structured or unstructured grids, meshing fluid or solid domains. The method produces robust transformations (Gillebaart *et al.*, 2016) and high quality meshes (Aubert *et al.*, 2017; Gillebaart *et al.*, 2016).

However, for large problems with huge meshes, the method becomes very expensive in CPU time for both the weight calculation and mesh propagation steps. Improvements have been proposed to speed up the propagation step through a memory formulation and parallelization (Gillebaart *et al.*, 2016). The weight calculation speed-up has been achieved through the reduction of the number of control points with a *greedy algorithm* (Aubert *et al.*, 2017; Gillebaart *et al.*, 2016; Jakobsson and Amoignon, 2007; Rendall and Allen, 2009). This algorithm builds a quasi-optimal subset of control points used to morph the mesh, reducing considerably the propagation step. However, the selection of control points is still a bottleneck because it consists in a repetition of sequential operations (Gillebaart *et al.*, 2016). We propose a new method using the Schur complement and pre-assembling method to speed up the greedy step. A rib shape optimization based on a GA and kriging metamodels is proposed to evaluate the morphing performances (Simpson *et al.*, 2001). The bi-objective problem is to maximize the heat transfer, expressed through the Nusselt number, while minimizing the head loss. These performances are obtained using CFD simulations of several periodic ribs. The shape changes are performed using morphing and re-meshing.

In Section 2, the RBF-based mesh morphing formulation is introduced and the improvements are presented in Section 3. The mesh morphing method is then applied to a rib shape optimization and the results are compared with re-meshing method ones in Section 4.

2. Mesh morphing formulation

2.1 Weights calculation

RBF-based mesh deformation is a node-by-node scheme free from mesh connectivity. It propagates by interpolation the known motion of a cloud of control points to the mesh nodes. Usually, control points are located on mesh boundaries. The interpolated function is the displacement $\vec{s}$ of a node, defined as the weighted sum of RBFs (Aubert *et al.*, 2017) by:

$$\vec{s}(\vec{x}) = \sum_{j=1}^{N_c} \phi\left(d\left(\vec{x} - \vec{x}_{cj}\right)\right)\vec{\gamma}_j \tag{1}$$

where $\vec{x}$ is the initial (i.e. before deformation) node position, N_c is the number of control points, ϕ is the chosen RBF, d is the chosen function measuring distance between two positions, $\vec{x}_{cj}$ and $\vec{\gamma}_j$ are respectively the initial position and weight of the j-th control point.

It should be noticed that only initial positions appear explicitly in equation (1). The prescribed displacement of control points and other constraints are embedded in the weights $\{\vec{\gamma}_i\}_{1 \le i \le N_c}$. When these are known, $\vec{s}$ is fully defined.

As proposed by Aubert *et al.* (2017), the set of control points is split into two ordered subsets, composed of moving control points (from 1 to N_m) followed by sliding control points (from $N_m + 1$ to $N_m + N_s = N_c$) Different conditions are associated with each subset (Aubert *et al.*, 2017):

- the displacement $\overrightarrow{s}$ must satisfy the interpolation condition for each moving control point:

$$\forall i \in [1, N_m], \quad \sum_{j=1}^{N_c} \phi\left(d\left(\overrightarrow{x}_{mi} - \overrightarrow{x}_{cj}\right)\right) \overrightarrow{\gamma}_j = \overrightarrow{s}_{mi} \tag{2}$$

where $\overrightarrow{s}_{mi}$ is the known displacement of the i-th moving point initially at the position $\overrightarrow{x}_{mi}$.

- $\overrightarrow{s}$ the normal component $\overrightarrow{s} \cdot \overrightarrow{n}$ must be zero for each sliding control point:

$$\begin{aligned} \forall i \in \quad & [N_m + 1, N_m + N_s], \\ & \sum_{j=1}^{N_c} \phi\left(d\left(\overrightarrow{x}_{si} - \overrightarrow{x}_{cj}\right)\right) \overrightarrow{\gamma}_j \cdot \left\{\sum_{k=1}^{3} n_k \overrightarrow{e}_k\right\} = 0 \end{aligned} \tag{3}$$

where $\overrightarrow{x}_{si}$ is the initial position of the i-th sliding point.

- none of the sliding control points contributes to the displacement in the tangential directions $\overrightarrow{t}$ and $\overrightarrow{b}$:

$$\forall i \in [N_m + 1, N_m + N_s], \overrightarrow{\gamma}_i \cdot \overrightarrow{t} = 0 \tag{4}$$

$$\overrightarrow{\gamma}_i \cdot \overrightarrow{b} = 0 \tag{5}$$

Therefore, the components of $\{\overrightarrow{\gamma}_i\}_{1 \le i \le N_c}$ are the solution of the $(3N_c) \times (3N_c)$ linear system:

$$\begin{pmatrix} \Phi^m & 0_{m\times c} & 0_{m\times c} \\ 0_{m\times c} & \Phi^m & 0_{m\times c} \\ 0_{m\times c} & 0_{m\times c} & \Phi^m \\ n_1\Phi^s & n_2\Phi^s & n_3\Phi^s \\ t_1\Phi^t & t_2\Phi^t & t_3\Phi^t \\ b_1\Phi^t & b_2\Phi^t & b_3\Phi^t \end{pmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ 0_s \\ 0_s \\ 0_s \end{pmatrix} \tag{6}$$

where

- the left-hand side matrix is called the *interpolation matrix*.
- the matrices Φ^m, Φ^s and Φ^t are defined as:

$$\Phi^m = \left[\phi\left(d\left(\overrightarrow{x}_{ci} - \overrightarrow{x}_{cj}\right)\right)\right]_{1 \le i \le N_m, 1 \le j \le N_c} \tag{7}$$

$$\Phi^s = \left[\phi\left(d\left(\vec{x}_{c(N_m+i)} - \vec{x}_{cj}\right)\right)\right]_{1 \le i \le N_s, 1 \le j \le N_c} \tag{8}$$

$$\Phi^t = \begin{pmatrix} 0_{s\times m} & I_{s\times s} \end{pmatrix} \in \mathbb{R}^{N_s \times N_c} \tag{9}$$

- $0_{m \times c}$ is the null matrix in $\mathbb{R}^{N_m \times N_c}$, $0_{s \times m}$ is the null matrix in $\mathbb{R}^{N_s \times N_m}$ and $I_{s\times s}$ is the identity matrix in $\mathbb{R}^{N_s \times N_s}$.
- vectors Γ_k and S_k are defined as:

$$\forall k \in [1,3], \quad \Gamma_k = \left[\vec{\gamma}_i \cdot \vec{e}_k\right]_{1 \le i \le N_c} \tag{10}$$

$$S_k = \left[\vec{s}_{mi} \cdot \vec{e}_k\right]_{1 \le i \le N_m} \tag{11}$$

- 0_s is the null vector in $\mathbb{R}^{N_s}$.
- (n_1, n_2, n_3), (t_1, t_2, t_3) and (b_1, b_2, b_3) are respectively the components of $\vec{n}$, $\vec{t}$ and $\vec{b}$ in $(\vec{e}_1, \vec{e}_2, \vec{e}_3)$ basis.

2.2 Mesh morphing

In matrix notation, equation (1) becomes:

$$S^N = \Phi \Gamma^N \tag{12}$$

where the displacement matrix S^N, the matrix of dependence from nodes to control points Φ and the weights matrix Γ^N are defined as:

$$S^N = \left[\vec{s}(\vec{x}_i) \cdot \vec{e}_k\right]_{1 \le i \le N, 1 \le k \le 3} \tag{13}$$

$$\Phi = \left[\phi\left(d\left(\vec{x}_i - \vec{x}_{cj}\right)\right)\right]_{1 \le i \le N, 1 \le j \le N_c} \tag{14}$$

$$\Gamma^N = \left[\vec{\gamma}_i \cdot \vec{e}_k\right]_{1 \le i \le N_c, 1 \le k \le 3} \tag{15}$$

Several operations are necessary to morph a mesh as shown in Algorithm 1 (Alg. 1). Firstly, the linear system defined by equation (6) is assembled and solved to obtain the weights (Alg. 1-lines 1-1). A lower-upper (LU) factorization (Press *et al.*, 1992) is used: a LU decomposition of the interpolation matrix is performed (*LU factor* step) and the system is solved using this decomposition (*LU solve* step). Secondly, the node displacements are interpolated for all the N mesh nodes by a simple matrix-matrix product with a prior assembly of Φ (Alg. 1-line 1). Finally, this is simply added to the initial node positions $\vec{x}_i$ to obtain the final positions $\vec{x}_i^{\,*}$ (Alg. 1-line 1). The computational cost of solving equation (6) scales with $\mathcal{O}\left(\frac{2}{3}(3N_c)^3 + \frac{1}{2}(3N_c)^2\right)$ for the *LU factor* and $\mathcal{O}\left(2\,(3N_c)^2\right)$ for the *LU solve*. The interpolation and update steps scale with $\mathcal{O}(6N \times N_c)$ and $\mathcal{O}(3N)$, respectively.

Algorithm 1: Mesh morphing
Input: Initial position of nodes: $X = \{\vec{x}_i\}_{1 \le i \le N}$
Input: Initial position of control points: $\{\vec{x}_{ci}\}_{1 \le i \le N_c}$
Input: Displacement of moving points: $\{\vec{s}_{mi}\}_{1 \le i \le N_m}$
Input: Matrix of dependence: Φ
Output: Final position of nodes: $X^* = \{\vec{x}_i{}^*\}_{1 \le i \le N}$
Compute the weights $\{\vec{\gamma}_i\}_{1 \le i \le N_c}$ *:*
Assemble the interpolation matrix and right-hand side (RHS) of equation (6)
Solve equation (6) (*LU factor* and *LU solve*) to get (Γ_1, Γ_2, Γ_3)
Interpolate the nodes displacement:
$S^N = \Phi\Gamma^N$
Update the nodes position:
$X^* = X + S^N$

3. Improvement of the greedy algorithm efficiency

The greedy algorithm builds iteratively a quasi-optimal subset of control points by adding to the actual subset the control point with maximum interpolation error [Alg. 2 in (Aubert *et al.*, 2017)]. The displacement field, computed using only the control points of the current subset, is applied to move all the provided control points, using Alg. 1. The displacement of the control points in this subset is exactly interpolated. So, one control point not already chosen is exhibiting the largest error. If its error is greater than some tolerance, usually defined as a fraction of $s_{\max}$, it is added to the subset and the whole process is repeated. An improved formulation based on the iterative resolution and pre-assembling method is detailed in Alg. 2 of the present work.

Algorithm 2: Improved greedy algorithm
Input: Initial position of nodes: $X = \{\vec{x}_i\}_{1 \le i \le N}$
Input: Displacement of moving points: $\{\vec{s}_{mi}\}_{1 \le i \le N_m}$
Input: Number of points before restarting: N_{res}
Output: Final subset of control points of size N_g
Initialize the subset of control points and its initial size N_g
Build the matrix Φ^c using equation (16)
Using Alg. 1 move all the control points and store the LU decomposition of the interpolation matrix as $\Phi^{\bullet}_{LU}$ and weights as (Γ^g)
$\forall\, i \in [1, N_m]$ eval $\|(\vec{x}_{mi}{}^* - \vec{x}_{mi}) - \vec{s}_{mi}\|$
$\forall\, i \in [1, N_m]$ eval $|(\vec{x}_{si}{}^* - \vec{x}_{si}) \cdot \vec{n}|$
Find the point with the largest *error*
repeat /* Greedy loop */
  $N_a = 0$
  repeat /* Iterative resolution loop */
    if *error* > *tolerance* **then**
      Add this point to the control points subset
      $N_a = N_a + 1$
    end
    Compute block matrices and RHS of equation (24):
      Assemble Φ°_a, $\Phi^{\triangledown}$ and $\Phi^{\triangle}$ using Φ^c
      Assemble S^a
    Using equations (28) and (29) get Γ^a and Γ^g
    Find the point with largest error:

```
17          Move all the control points using Alg. 1 with Φ^c, Γ^g and Γ^a
18          ∀i ∈ [1, N_m], eval ||(x⃗_mi* − x⃗_mi) − s⃗_mi||
19          ∀i ∈ [1, N_s], eval |(x⃗_si* − x⃗_si) · n⃗|
20          Find the point with the largest error
21      until (N_a ≥ N_res) or (error ≤ tolerance);
22      N_g = N_g + N_a
        Compute the restart step:
23          Re-assemble Φ• from block matrices [Equation (24)]
24          Compute LU factor of Φ• to get Φ•_LU
25          Re-assemble (Γ^g) from the vectors Γ^g and Γ^a
26  until error ≤ tolerance;
```

3.1 Pre-assembling method

Before the greedy loop starts (Alg. 2-line 7), the matrix Φ^c containing the ϕ evaluations is built:

$$\Phi^c = \left[\phi\left(d\left(\vec{x}_{ci} - \vec{x}_{cj}\right)\right)\right]_{1 \le i \le N_c, 1 \le j \le N_c} \tag{16}$$

During the motion of the control points as shown Alg. 2-lines 3 and 17, the interpolation matrix is assembled by re-indexing the values of Φ^c according to the control points in the subset. So, the number of ϕ evaluations is $N_c{}^2$ regardless of the number of greedy iterations. As the re-indexing cost is negligible compared with that for the ϕ evaluation, much work is saved. The number of control points is usually less than a few thousand, thus the storage size of Φ^c is estimated as less than 300 mb (Gillebaart *et al.*, 2016).

3.2 Iterative resolution

Using a greedy algorithm, the interpolation matrix and RHS of equation (6) grow iteratively. With the standard algorithm (Aubert *et al.*, 2017), the entire linear system is solved at each greedy iteration. With numerous points in the subset, this step is time-consuming. A resolution taking advantage of the iterative building is proposed to speed up the greedy process.

At each greedy iteration, the linear system of equation (6), written in matrix notation as:

$$\Phi^{\bullet}(\Gamma^g)' = S^g \tag{17}$$

Is solved to get the weights $(\Gamma^g)'$ of the N_g control points in the subset. When N_g is large, the interpolation matrix is large and the system resolution is time-consuming. The main idea of the iterative resolution is to take advantage of the LU decomposition and the actual solution of equation (17) to solve the next iteration.

When N_a new control points $\{\vec{x}_{ca}\}_{1 \le a \le N_a}$ are added to the previous subset of control points $\{\vec{x}_{cg}\}_{1 \le g \le N_g}$, new conditions for the weights calculation are added to the interpolation matrix, and the previous ones are updated. The new conditions are:

$$\forall a \in [1, N_a], \quad \underbrace{\sum_{j=1}^{N_g} \phi\left(d\left(\vec{x}_{ma} - \vec{x}_{cj}\right)\right)}_{\Phi_\nabla} \vec{\gamma}_j + \underbrace{\sum_{j=1}^{N_a} \phi\left(d\left(\vec{x}_{ma} - \vec{x}_{cj}\right)\right)}_{\Phi^\circ} \vec{\gamma}_j = \vec{s}_{ma} \tag{18}$$

for moving control points and

$$\forall a \in [1, N_a], \quad \underbrace{\sum_{j=1}^{N_g} \phi\left(d\left(\vec{x}_{sa} - \vec{x}_{cj}\right)\right)}_{\Phi_\nabla} \vec{\gamma}_j \cdot \left\{\sum_{k=1}^{3} n_k \vec{e}_k\right\} + \underbrace{\sum_{j=1}^{N_a} \phi\left(d\left(\vec{x}_{sa} - \vec{x}_{cj}\right)\right)}_{\Phi^\circ} \vec{\gamma}_j \cdot \left\{\sum_{k=1}^{3} n_k \vec{e}_k\right\} = 0 \tag{19}$$

$$\vec{\gamma}_a \cdot \vec{t} = 0 \tag{20}$$

$$\vec{\gamma}_a \cdot \vec{b} = 0 \tag{21}$$

for sliding ones. The updated conditions for moving control points are:

$$\forall g \in [1, N_g], \quad \underbrace{\sum_{j=1}^{N_g} \phi\left(d\left(\vec{x}_{mg} - \vec{x}_{cj}\right)\right)}_{\Phi^\bullet} \vec{\gamma}_j + \underbrace{\sum_{j=1}^{N_a} \phi\left(d\left(\vec{x}_{mg} - \vec{x}_{cj}\right)\right)}_{\Phi^\triangle} \vec{\gamma}_j = \vec{s}_{mg} \tag{22}$$

and for sliding control points:

$$\forall g \in [1, N_g], \quad \underbrace{\sum_{j=1}^{N_g} \phi\left(d\left(\vec{x}_{sg} - \vec{x}_{cj}\right)\right)}_{\Phi^\bullet} \vec{\gamma}_j \cdot \left\{\sum_{k=1}^{3} n_k \vec{e}_k\right\} + \underbrace{\sum_{j=1}^{N_a} \phi\left(d\left(\vec{x}_{sg} - \vec{x}_{cj}\right)\right)}_{\Phi^\triangle} \vec{\gamma}_j \cdot \left\{\sum_{k=1}^{3} n_k \vec{e}_k\right\} = 0 \tag{23}$$

So, new lines and columns are added to the interpolation matrix and new lines to the weight vector and RHS of the linear system in equation (17). The components of $\{\overrightarrow{\gamma}_i\}_{1 \le i \le (N_g+N_a)}$ are the solution of the block system:

$$\begin{pmatrix} \Phi^{\bullet} & \Phi^{\triangle} \\ \Phi^{\triangledown} & \Phi^{\circ} \end{pmatrix} \begin{pmatrix} \Gamma^g \\ \Gamma^a \end{pmatrix} = \begin{pmatrix} S^g \\ S^a \end{pmatrix} \tag{24}$$

where S^a is the RHS associated with the control points $\{\overrightarrow{x}_{ca}\}_{1 \le a \le N_a}$, Φ° the $\{\overrightarrow{x}_{ca}\}_{1 \le a \le N_a}$ self interaction conditions, $\Phi^{\triangledown}$ and $\Phi^{\triangle}$ the interaction conditions of $\{\overrightarrow{x}_{ca}\}_{1 \le a \le N_a}$ with the previous control points of the subset $\{\overrightarrow{x}_{cg}\}_{1 \le g \le N_g}$. Γ^a is the weight vector associated with $\{\overrightarrow{x}_{ca}\}_{1 \le a \le N_a}$ and Γ^g is the vector of the updated weights associated with $\{\overrightarrow{x}_{cg}\}_{1 \le g \le N_g}$, modified because of the newly added points.

The system equation (24) is rewritten as:

$$\Phi^{\bullet}\Gamma^g + \Phi^{\triangle}\Gamma^a = S^g \tag{25}$$

$$\Phi^{\triangledown}\Gamma^g + \Phi^{\circ}\Gamma^a = S^a \tag{26}$$

Introducing the Schur complement of size $(3N_a \times 3N_a)$ (Cottle, 1974) as:

$$Sch = \Phi^{\circ} - \Phi^{\triangledown}M \tag{27}$$

with M the solution of $\Phi^{\bullet}M = \Phi^{\triangle}$, two steps are required to compute the weights. Solving the linear system:

$$Sch\,\Gamma^a = S^a - \Phi^{\triangledown}(\Gamma^g)' \tag{28}$$

leads to Γ^a and then Γ^g is computed as the solution of the linear system:

$$\Phi^{\bullet}\Gamma^g = S^g - \Phi^{\triangle}\Gamma^a \tag{29}$$

The iterative resolution uses the LU decomposition of Φ° from the previous step to compute the matrix M and to solve equation (29). The advantage of solving the linear system using the Schur complement is that LU decompositions (*LU factor* step) and applications (*LU solve* step) are performed repeatedly only on smaller matrices of size $(3N_a \times 3N_a)$ to solve equation (28) compared with $(3N_g \times 3N_g)$ to solve equation (17). To maximize the benefits of this formulation, the main idea is to add several points N_a at a time, to re-build matrices Φ°, $\Phi^{\triangledown}$ and $\Phi^{\triangle}$, and to apply equations (28) and (29) as shown by the iterative loop Alg. 2-line 9. When N_a is greater than a prescribed number of control points N_{res}, a *restart step* is performed: a new matrix $\Phi^{\bullet}$ is assembled and its LU decomposition computed (Alg. 2-lines 23-25).

3.3 Optimal number of points before restarting

The performance of Alg. 2 lies with N_{res}. If it is too small, numerous re-assembling and LU decomposition of $\Phi^{\bullet}$ are necessary. If N_{res} is too large, the LU decomposition of Schur matrices of increasing size is computed. In both cases, the performances are the same as if there is no iterative resolution loop. Thus, the value of N_{res} has to be a trade-off between the

number of LU decompositions performed and the cost of these decompositions, which depends on the matrix size. So, N_{res} is defined as a fraction of $\Phi^{\bullet}$ size.

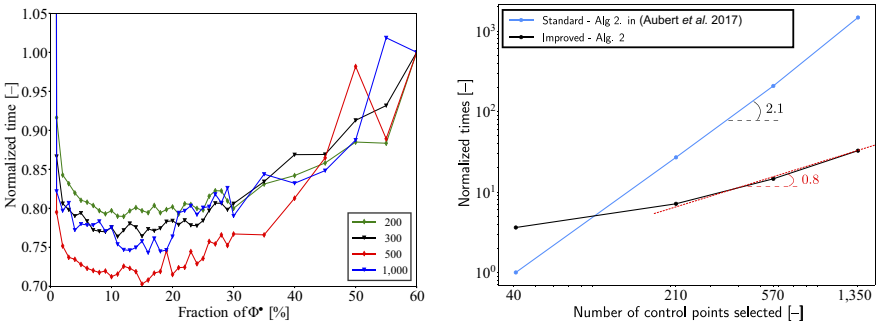
Figure 1(a) shows CPU times of a benchmark used to estimate the best value of N_{res}. The size of the final subset of control points N_g is an input of the test and is set to 200, 300, 500 and 1,000. Fractions of $\Phi^{\bullet}$ size in the range of 1 per cent to 60 per cent are chosen to solve the same problem. Time is normalized with the one necessary to solve the problem when N_{res} is 60 per cent of $\Phi^{\bullet}$.

The best speed up is achieved for N_{res} between 10 per cent and 20 per cent regardless of the problem size. The gain using Alg. 2 increases with the problem size until 500 control points and then decreases for 1,000 control points. A memory cache effect is suspected for the largest problem. The matrix cannot be stored in the memory cache, and time is lost in memory communication.

The improved Alg. 2 and standard greedy algorithms Alg. 2 in (Aubert *et al.*, 2017) are applied to the fan tip case presented by Aubert *et al.* (2017). N_{res} is set to 10 per cent of $\Phi^{\bullet}$ size. Figure 1(b) shows the execution time of both algorithms for different greedy tolerances (Alg. 2-line 10), which lead to select 41 to 1357 control points over 13,000 available. The CPU time scales as $N_c^{0.8}$ compared with $N_c^{2.1}$. At the beginning of the improved algorithm, the matrix e^c (size 13,000 × 13,000) is pre-computed. This step is performed regardless of the number of control points added by the greedy loop. So, for only 41 control points, the time needed to calculate this matrix leads to poorer performances considering the global efficiency of the method. Thus, the improvement is only significant for larger problems, as shown by Figure 1(b).

4. Application to rib shape optimization

The rib chosen for the optimization has a symmetric trapezoidal shape, as shown in Figure 2(a). The optimization parameters are the height (h_{rib}), the base width (E) and the top width (e) of the rib. The canal height (H) and the rib-pitch ($L = 1.8H$) remain constant. The angle θ is only used for the analysis. The are either two objectives: maximizing the Nusselt number Nu and minimizing the head loss coefficient C_f. Formally, the bi-objective problem is written as:



Notes: (a) Benchmark normalized time for the greedy loop over the fraction of Φ· size for different problem sizes: 200 (green), 300 (black), 500 (red) and 1,000 (blue) control points; (b) execution time of standard greedy (Alg. 2 in (Aubert *et al.*, 2017), blue) and improved (Alg. 2, dark) algorithms as a function of the number of control points selected (log-log scale)

Figure 1.
Improved algorithm performances benchmark

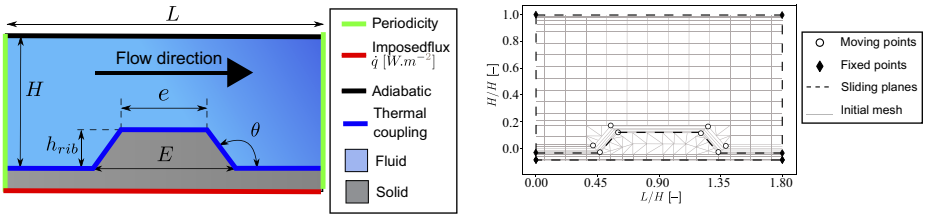


Figure 2.
Case setup

Notes: (a) Rib shape parameters and boundary conditions for CFD simulations; (b) control points definition for the morphing step

$$\text{maximize} \quad Nu\left(\overrightarrow{p}\right) \tag{30}$$

$$\text{minimize} \quad C_f\left(\overrightarrow{p}\right) \tag{31}$$

$$\text{with} \quad \overrightarrow{p} \in \mathcal{D} \tag{32}$$

$$\text{subject to} \quad h_{rib}/H \in [0.05; 0.3] \tag{33}$$

$$e/2H \in [0.045; 0.63] \tag{34}$$

$$E/2H \in [0.0405; 0.72] \tag{35}$$

$$E \geq 0.9\,e \tag{36}$$

$$E \leq 0.9\,e + 0.35 \tag{37}$$

with $\overrightarrow{p}$ the vector of parameters and $D = \{h_{rib}, e, E \mid \text{equations (33)-(37)}\}$ the constrained parameters space. The constraint equation (36) ensures that the shapes remain manufacturable. Shapes exceeding the constraint equation (37) lead to degenerate meshes using the present morphing setup. The solution of the problem is the best trade-off between Nu and C_f, called the *Pareto front*.

The problem is solved using the genetic algorithm NSGA-II (Deb *et al.*, 2002) together with kriging-based metamodels (Simpson *et al.*, 2001). Two kind of metamodels, built using a full filled design of experiments (DOE) of 227 points, are compared. The first ones are built using re-meshing and the seconds are using the present morphing method.

4.1 Case setup

4.1.1 CFD setup. Code_Saturne software (EDF R&D, 2017) is used to solve the Navier–Stokes equations for two-dimensional unsteady air flow and heat transfer in the rib channel. The simulation is performed in dimensionless form, with all lengths made dimensionless by the channel height. The $BL - \overline{v^2}/k$ model is used to model the turbulence (Billard and

Laurence, 2012). This wall-resolved model takes into account the turbulence anisotropy ($\overline{v^2}$) in the wall normal direction. Thus, it is recommended for heat transfer problems with recirculating flows such as ribs (Billard and Laurence, 2012; Kumar and Kumar, 2017). The boundary conditions are shown in Figure 2(a). The fluid and solid domains are thermally coupled (blue line) using a monolithic formulation. The flow is assumed spatially developed, thus a streamwise periodic condition is applied (green line). The top wall is adiabatic (black line) and an imposed heat flux density $\dot{q}$ is applied on the bottom wall (red line). The Reynolds number, based on the hydraulic diameter $D_H = 2H$, is set to 20,000. The Nusselt number is calculated as:

$$Nu = \frac{\dot{q}\, D_H}{\lambda\,(T_{wall} - 0.5(T_{in} + T_{out}))} \tag{38}$$

with T_{wall} the mean surface temperature at the bottom wall, T_{in} and T_{out} the bulk temperature at the inlet and outlet. The head loss coefficient is calculated as:

$$C_f = \frac{\Delta P_{in,out}}{0.5\rho {U_{deb}}^2}\frac{D_H}{L} \tag{39}$$

with $\Delta P_{in,out}$ the pressure drop between inlet and outlet and U_{deb} the bulk velocity. Performances are normalized using Dittus–Boettler Nu_0 and Blasius C_{f_0} correlations for smooth channels (Rau *et al.*, 1998).

The mesh is an unstructured grid with about 160,000 cells mixing hexahedral and prismatic elements. Both fluid and solid domains are meshed. A boundary layer mesh is used to satisfy the $y^+ = 1$ condition for the first cell near the wall, as required by the turbulence model.

4.1.2 Morphing setup. The scaled unbiased Euclidean norm is used to measure the distance between two positions:

$$d(\overrightarrow{u}) = \frac{1}{r_0}||\overrightarrow{u}|| \tag{40}$$

where r_0 is referenced as the support radius. Wendland's C^2 compactly-supported function is chosen for its efficiency in CFD applications (Rozenberg *et al.*, 2014):

$$\phi(d) = \begin{cases} (4d+1)(1-d)^4, & \text{if } d < 1 \\ 0, & \text{if } d \geq 1 \end{cases} \tag{41}$$

ϕ is monotonically decaying from a control point as d increases, and it is null if the scaled distance is larger than 1.

As shown in Figure 2(b) only the four corners of the rib and four points at the outer edge of the boundary layer mesh are defined as moving points (°) with a prescribed displacement function of $\overrightarrow{p}$. The channel corners are fixed points (◆) with a null displacement. A total of 3,213 nodes on dashed lines are set as sliding points. The support radius is set to four times the largest displacement $s_{\max} = max\left(||\delta\overrightarrow{p}||\right)$ with $\delta\overrightarrow{p}$ the variation between the initial parameters and the new ones. The initial parameters are $h_{rib}/H = 0.15$, $e/2H = 0305$ and $E/$

$2H = 0.435$ The greedy tolerance, set to $1.10^{-5}s_{\max}$, is achieved using all the moving and fixed points and 300 sliding points (on average). N_{res} is set to 10 per cent.

4.1.3 Data analysis. A clustering method based on a Gaussian mixture (Bishop, 2006) is used to define centers, highlighting a finite number of optimal shapes.

4.2 Results

One of the major challenges with morphing is to ensure the mesh quality after deformation. The mesh comparison at the top of Figure 3 confirms that RBF-based morphing preserves the boundary layer mesh and re-meshing (Aubert *et al.*, 2017), but the prismatic cells at the left are severely deformed. However, the cell center offset metric highlights an equivalent mesh quality between the two methods: only 3 per cent of the cells have a metric value less than the 0.2. This metric, computed by *Code_Saturne*, ensures solver stability and an accurate gradient calculation. This proportion is nonetheless not prejudicial to the resolution.

As the re-meshing method creates a new mesh, each simulation has to start from a uniform solution. The use of morphed meshes allows calculation restarts without any data interpolation as the mesh connectivity is preserved. Thus only one simulation starting from a uniform solution is required. Using calculation restart instead of uniform initialization leads to a speed-up of the resolution of about 40 per cent. So to perform the 227 DOE shapes simulations, the total resolution time using re-meshing is almost twice as the fact that using morphing.

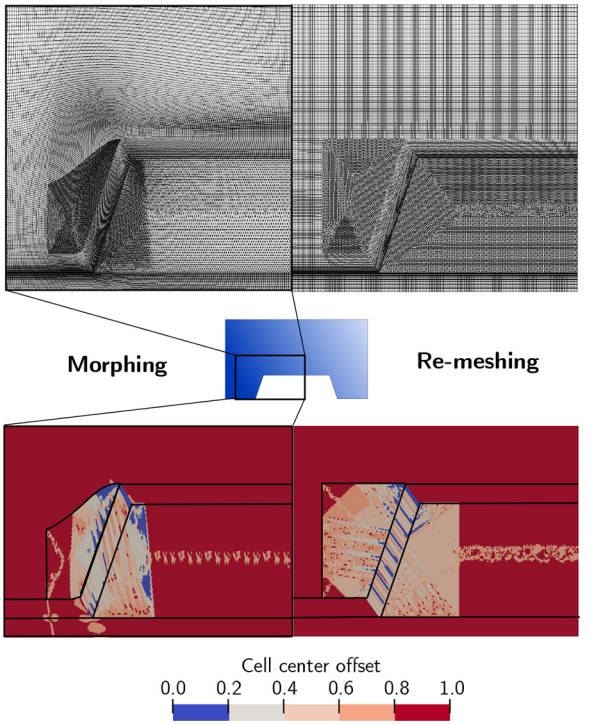
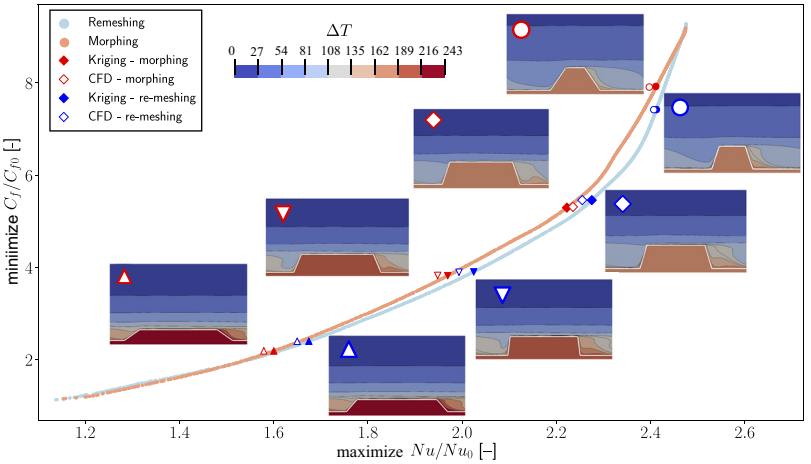


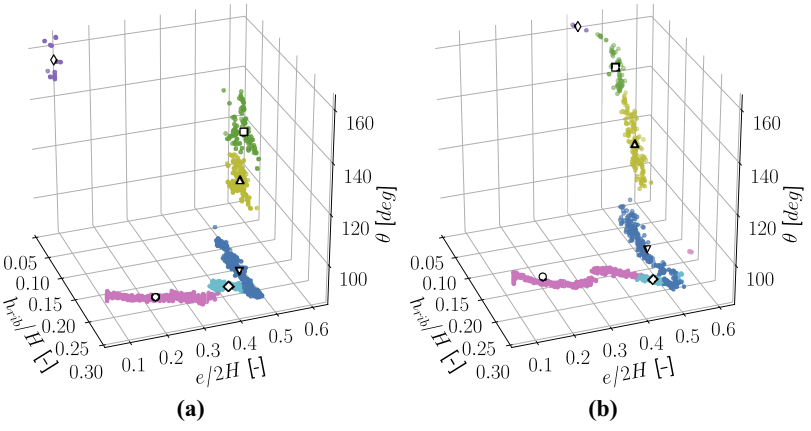
Figure 3.
Comparison of the influence of re-meshing and morphing strategies on mesh quality for a rib shape of the Pareto front (◇)

The two Pareto fronts achieved using the two metamodels are quite similar, as can be seen in Figure 4. The clustering of the population distribution in Figure 5 leads to the same conclusion. The constraint equation (37), which limits the exploration of D to avoid degenerated morphed meshes, has no influence on the achieved Pareto front. Thus, the optimization result is not related to the meshing strategy and is quite robust.



Notes: The re-meshing based metamodel results (blue line) are compared to the morphing-based ones (red line). Filled symbols represent the cluster centers performances predicted by the metamodels. Open symbols are the performance calculated from reconverged CFD simulation. Temperature variations associated to four optimal designs are superimposed

Figure 4. Pareto front and cluster centers achieved using the two metamodels.



Notes: (a) Re-meshing based metamodel; (b) morphing-based metamodel

Figure 5. Population distribution in $\mathcal{D}$ with clusters and their centers

The clustering gives four interesting centers in both cases and their performances predicted from the metamodels are shown in Figure 4 as filled symbols. The four shapes highlighted are similar between morphing and re-meshing. CFD runs were performed to qualify the metamodel interpolation on these centers, which appear as empty symbols on Figure 4 with the associated temperature variation fields. The Nu difference between the CFD and the metamodel is 0.87 ± 0.6 per cent for morphing and 1.05 ± 0.9 per cent for re-meshing. For C_f, it is about 0.17 ± 0.13 per cent for morphing and 0.07 ± 0.14 per cent for re-meshing. The interpolation accuracy of the morphing-based metamodel is slightly better for the Nusselt number. The two interpolation accuracies are equivalent for the head loss coefficient. Thus, morphing-based simulations lead to a slightly better metamodel. Again, this can probably be explained by the preservation of mesh connectivity. Indeed, it is well-known that the mesh has a significant influence on the CFD results. As the mesh connectivity does not change with morphing, the objectives variations may be evaluated with fewer uncertainties. The CFD response is thus more continuous and easier to interpolate.

The four shapes highlighted by the clustering are similar for morphing and re-meshing. Shapes with a maximal rib height (° and ◇) lead to large heat transfer associated with high pressure drop. A slim rib (°) induces a large recirculation area and increases Nu and C_f. When the rib height is smaller and for significant θ angles (△), C_f is smaller and the temperature stratification reduces the heat transfer. These results highlight the major influence of the rib height h_{rib} on the heat transfer and pressure drop as reported by Kim *et al.* (2010).

5. Conclusions

An improved greedy algorithm using pre-assembling method and iterative resolution has been described. As the greedy algorithm adds one control point at a time, the linear system inherent to RBF-based morphing is build iteratively using a four blocks matrix. This system is solved using the Schur complement and the solution of the previous iteration. It appears that the number of control points to add before restarting is a key parameter of the method. A value between 10 per cent and 20 per cent of the previous iteration matrix size is found to minimize the resolution time.

The method was then applied to the shape optimization of a rib. The comparison between re-meshing and morphing-based metamodels has highlighted the benefits of using morphing: less simulation time and slightly more accurate metamodels (according to the CFD). However, the entire parameters space is not attainable with this method.

Further works will focus on using different initial meshes with intersecting attainable areas to cover the entire parameters space.

References

Aubert, S., Mastrippolito, F., Rendu, Q., Buisson, M. and Ducros, F. (2017), "Planar slip condition for mesh morphing using radial basis functions", *Procedia Engineering*, Vol. 203, pp. 349-361.

Batina, J.T. (1990), "Unsteady Euler airfoil solutions using unstructured dynamic meshes", *AIAA Journal*, Vol. 28 No. 8, pp. 1381-1388.

Billard, F. and Laurence, D. (2012), "A robust $k-\varepsilon-\overline{v^2}/k$ elliptic blending turbulence model applied to near-wall, separated and buoyant flows", *International Journal of Heat and Fluid Flow*, Vol. 33 No. 1, pp. 45-58.

Bishop, C. (2006), *Pattern Recognition and Machine Learning, Chapter 9 and 10*, Springer, New York, NY, pp. 423-450.

Cottle, R.W. (1974), "Manifestations of the Schur complement", *Linear Algebra and its Applications*, Vol. 8 No. 3, pp. 189-211.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002), "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, Vol. 6 No. 2, pp. 182-197.

EDF R&D (2017), "Code_saturne version 5-1", available at: www.code-saturne.org/

Gillebaart, T., Blom, D., van Zuijlen, A. and Bijl, H. (2016), "Adaptive radial basis function mesh deformation using data reduction", *Journal of Computational Physics*, Vol. 321, pp. 997-1025.

Jakobsson, S. and Amoignon, O. (2007), "Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization", *Computers & Fluids*, Vol. 36 No. 6, pp. 1119-1136.

Kim, K.M., Kim, B.S., Lee, D.H., Moon, H. and Cho, H.H. (2010), "Optimal design of transverse ribs in tubes for thermal performance enhancement", *Energy*, Vol. 35 No. 6, pp. 2400-2406.

Kumar, R. and Kumar, A. (2017), "Computational fluid dynamics based study for analyzing heat transfer and friction factor in semi-circular rib-roughened equilateral triangular duct", *International Journal of Numerical Methods for Heat and Fluid Flow*, Vol. 27 No. 4, pp. 941-957.

Liu, J., Xie, G., Sunden, B.A., Wang, L. and Andersson, M. (2017), "Enhancement of heat transfer in a square channel by roughened surfaces in rib-elements and turbulent flow manipulation", *International Journal of Numerical Methods for Heat and Fluid Flow*, Vol. 27 No. 7, pp. 1571-1595.

Luo, L., Du, W., Wang, S., Wu, W. and Zhang, X. (2018), "Multi-objective optimization of the dimple/protrusion channel with pin fins for heat transfer enhancement", *International Journal of Numerical Methods for Heat and Fluid Flow*, Vol. 29 No. 2, pp. 790-813.

Mohammadi, B. and Pironneau, O. (2004), "Shape optimization in fluid mechanics", *Annual Review of Fluid Mechanics*, Vol. 36 No. 1, pp. 255-279.

Nielsen, E.J. and Anderson, W.K. (2002), "Recent improvements in aerodynamic design optimization on unstructured meshes", *AIAA Journal*, Vol. 40 No. 6, pp. 1155-1163.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. and Kramer, P.B. (1992), "Numerical recipes in C – the art of scientific computing", *Chap. 2 – Solution of Linear Algebraic Equations*, 2nd ed., Cambridge Univ Press, pp. 43-50.

Rau, G., Cakan, M., Moeller, D. and Arts, T. (1998), "The effect of periodic ribs on the local aerodynamic and heat transfer performance of a straight cooling channel", *Journal of Turbomachinery*, Vol. 120 No. 2, pp. 368-375.

Rendall, T. and Allen, C. (2009), "Efficient mesh motion using radial basis functions with data reduction algorithms", *Journal of Computational Physics*, Vol. 228 No. 17, pp. 6231-6249.

Rozenberg, Y., Aubert, S. and Bénéfice, G. (2014), "Fluid structure interaction problems in turbomachinery using rbf interpolation and greedy algorithm", *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition, V02BT39A033–V02BT39A033*.

Sederberg, T.W. and Parry, S.R. (1986), "Free-form deformation of solid geometric models", *ACM SIGGRAPH Computer Graphics*, Vol. 20 No. 4, pp. 151-160.

Simpson, T.W., Poplinski, J., Koch, P.N. and Allen, J.K. (2001), "Metamodels for computer-based engineering design: survey and recommendations", *Engineering with Computers*, Vol. 17 No. 2, pp. 129-150.

Corresponding author
Franck Mastrippolito can be contacted at: franck.mastrippolito@doctorant.ec-lyon.fr