Deformation Exchange between Adjacent Physical Code Geometries

B. Duplex$^{(1,3)}$ G. Gesquière$^{(3)}$ M. Daniel$^{(3)}$ F. Perdu$^{(2)}$

$^{(1)}$ CEA, DEN
DER/SSTH/LDAL
Cadarache
Saint-Paul-lez-Durance
F-13108, France

benjamin.duplex@cea.fr

$^{(2)}$ CEA, DEN
DER/SSTH/LDAS
17, rue des Martyrs
Grenoble
F-38054, France

fabien.perdu@cea.fr

$^{(3)}$ LSIS UMR CNRS 6168
ESIL Case 925
163 Avenue de Luminy
Marseille cedex 9
F-13288, France

benjamin.duplex@lsis.org
gilles.gesquiere@lsis.org
marc.daniel@lsis.org

ABSTRACT

Physical codes are dependent on a discipline and describe the reality through equations solved on a geometry. Strong links, or couplings, between them are set up to increase the result accuracy, leading to data exchanges. Mechanical codes calculate geometrical deformations which must be tranferred by couplings. Their direct exchange is generally impossible if coupled codes compute with different models of the same geometry (in practice different meshes). Moreover, two meshes can only share border(s). This paper focuses on this case: one mesh must follow the border deformation of the mechanical code mesh.

Keywords

radial basis function, CS-RBF, physical code coupling, deformation exchange, experimental device

1 INTRODUCTION

The French Atomic Energy and Energy Alternatives Commission [1] (CEA) uses codes to simulate physical behaviours in nuclear reactors. They are specialized to a specific domain, such as mechanics, thermohydraulics or fuel behaviour. The current CEA approach is to couple codes to improve the final result accuracy, leading to data exchanges between codes. These exchanges can be of various forms: point values (*e.g.* a temperature at a point), data fields (*e.g.* a pressure on a surface) or geometrical deformations. A mechanical code calculates geometrical modifications. Previous works have been made on the data exchange problem ([BDS07, PV08]). In this paper, we focus on the transfer of deformations calculated by a mechanical code to a second one in the particular but very important case where code geometries only share border(s).

The code geometrical models could be CSG or meshes. In this work, only meshes are considered because it corresponds to the most encountered situation. Each code is based on a particular mesh to optimize its computations with specific dimension (*e.g.* surfacic or volumic), properties (*e.g.* regularity and uniformity) and cell types (*e.g.* triangles, quadrangles). Most of them are composed of hundreds of thousand cells up to millions of cells. It must be added that the mesh deformations must keep the computed data attached to the mesh for further applications, which is against a full remeshing.

Based on the differences between meshes, it appears useful to define a continous function to model the geometrical deformations. This function can easily be sampled to any point where a displacement must be computed. One of the constraint is to interpolate the deformations and not to approximate them. Another and natural constraint is that the method must have a lower processing time order than the code calculation: *e.g.* hours when the code execution time is in days (involving generally several hundred thousand cells).

[1] CEA: French Atomic Energy and Energy Alternatives Commission (`http://www.cea.fr/english_portal`)

The remainder of the paper is structured as follows: section 2 reviews related work. Section 3 presents our method and its results are proposed in section 4.

2 RELATED WORK

The problem is to transfer an interpolated deformation between meshes that only share border(s).

In [DGDP10], we addressed the problem of transferring the deformations between meshes modelling the same geometry. We introduced a function F_α computed on a network of radial basis functions (RBFs) after a mandatory simplification step to receive reasonable computing times. The RBF network is a space deformation function ([Wen05]), the deformed area being the same that the one where the network is computed. This approach is thus not directly applicable.

To address our specific problem, the methods must be decomposed into two steps: the deformation of the first area must be applied to the border of the second area and then the inner part of the second area must be processed subject to its border deformation. For the second step, different approaches exist. Full remeshing techniques ([FG08]) are possible. However, it involves the loss of the data attached to the former inner nodes. In [CSW09], the authors apply a mass-spring model. It changes the inside point position iteratively until the complete stabilization. This method is unfortunatly too expensive in our case. In [YK09], a smoothing method optimizes the aspect ratio and reduces the element skew of cells. However, some nodes initially placed inside could finally be outside the mesh to smooth if new borders are within the initial mesh. In [Hel03], a Laplacian method is employed. Results shown are good but deteriorated for large displacements. RBFs are used in [dBvdSB07]. They interpolate the variation of node positions located on the mesh border. RBFs involve to solve a linear system whose size depends on the number of nodes to interpolate. In our case, this number is very high. It induces a computational time that exceeds target set by the CEA. It is the reason why we introduced function F_α ([DGDP10]).

3 THE PROPOSED METHOD

3.1 Overview of function F_α

F_α combines a simplification method (not detailed in this paper but based on [GH97]) which keeps α nodes, and a RBF network which interpolates these points. Due to the simplification step, F_α does not interpolate the exact deformation. As the kept nodes are considered the most important ones according to the simplification method criterion, the induced error is very low. These nodes are called landmark points $\{L_i\}_{i=1,..,\alpha} \in \mathbb{R}^3$. For any point $P \in \mathbb{R}^3$, function $F_\alpha : \mathbb{R}^3 \to \mathbb{R}^3$ is defined by:

$$F_\alpha(P) = \sum_{i=1}^{\alpha} \lambda_i.\phi(\|P - L_i\|_2) \quad + \mathscr{L}(P)$$

where ϕ is the C^2 Thin-Plate Spline (TPS) such that $\phi(r) = r^4.log(r)$. They are centered on landmarks L_i. $\lambda_i \in \mathbb{R}^3$ are the network coefficients and $\mathscr{L}$ is an affine function corresponding to global displacements:

$$\mathscr{L}(P) = \sum_{c=1}^{3} \lambda_{\alpha+c}.P^{(c)} \quad + \lambda_{\alpha+4}$$

where $P^{(c)}$ is the c^{th} coordinate of point P. With the same notations, the additional constraints are imposed to take into account the polynomial terms of $\mathscr{L}$:

$$\sum_{i=1}^{\alpha} \lambda_i.L_i^{(c)} = \sum_{i=1}^{\alpha} \lambda_i = 0, \quad \forall c = 1,2,3$$

Centers $\{L_i\}_{i=1,..,\alpha}$ of TPSs are known value positions that function F_α must interpolate. If L_i (resp. L_i') denotes the i^{th} node of initial (resp. deformed) mesh used by the mechanical code, the interpolation condition is: $F_\alpha(L_i) = L_i'$; $i = 1,..,\alpha$. The network coefficients are thus calculated by solving a linear system involved by these equations. We define matrix $\Psi \in \mathbb{R}^{\alpha\times\alpha}$ by:

$$\Psi = \Big(\phi(\|L_i - L_j\|_2)\Big)_{1\le i,j\le\alpha}$$

and matrix ξ by $\xi = (x_i, y_i, z_i, 1)_{1\le i\le\alpha} \in \mathbb{R}^{4\times\alpha}$, with $(x_i, y_i, z_i) \in \mathbb{R}^3$ the coordinates of point L_i. Matrix $A \in \mathbb{R}^{(\alpha+4)\times(\alpha+4)}$ of the linear system induced by the RBF network is defined by:

$$A = \begin{pmatrix} \Psi & \xi \\ \xi^T & \mathbf{0} \end{pmatrix}; \quad \text{with } \mathbf{0} = (0)_{4,4} \in \mathbb{R}^{4\times4}$$

The RBF calculation using TPSs implies a dense matrix A, symmetric but with a zero-filled diagonal. Parameter α must be small enough to have acceptable computation time and memory space used, involving an error during the deformation calculation.

The disadvantages exposed above are due to two aspects of TPSs. Functions $\phi(r)$ are exponentially growing, global to the whole data and equal to zero when $r = 0$. Ψ is symmetric, because: $\forall 1 \le i,j \le \alpha$, $\phi(\|L_i - L_j\|_2) = \phi(\|L_j - L_i\|_2)$, but not positive definite since $\phi(\|L_i - L_i\|_2) = \phi(0) = 0$. A is dense and symmetric with zero diagonal. That prevents to use efficient methods for solving the linear system, like iterative methods.

3.2 Definition of a new function F_α^σ

In order to overcome the difficulties detailed in the previous section, we propose in this paper to use functions with compact support. Their advantage is to vanish when the distance between two centers L_i and L_j of the RBF exceeds a threshold, leading to spare matrices Ψ and A. According to [RA10], the most efficient functions are the CS-RBF, Compactly-Supported Radial

Basis Function, proposed by Wendland ([Wen05]). For meshes in $\mathbb{R}^3$, $\phi(r) = (1-r)^4.(4r+1)$ if $r < 1$, $\phi(r) = 0$ otherwise. This function is defined on interval $[0,1]$ with $\phi(0) = 1$ and $\forall r \in]0,1[,\ 0 < \phi(r) < 1$. This provides a matrix Ψ symmetric positive definite, sparse and diagonally dominant. However, this function requires a support size. Although works have already been focused on solving this problem ([OBS05]), it remains open.

We introduce parameter σ. For a given CS-RBF, we calculate the minimal radius s_i of the ball containing σ centers $\{L\}$. The global support size S is obtained to ensure that every CS-RBF involve at least σ centers (*e.g.* $S = \max_{1 \le i \le \alpha} s_i$). Its calculation is in $O(\alpha.log(\alpha))$ and is only done once. A new interpolation function, denoted F_α^σ, is defined. It corresponds to F_α ([DGDP10]) with TPS replaced by CS-RBF. Its support size is calculated via σ as detailed above.

3.3 Mesh deformation

Transferring deformations to a mesh only sharing borders with the one used by the mechanical code can be realized through F_α^σ. Our whole method is detailed in figure 1, every step being described below. To begin, a mesh is computed from the geometry of the whole device ($1a$). The mechanical code calculation

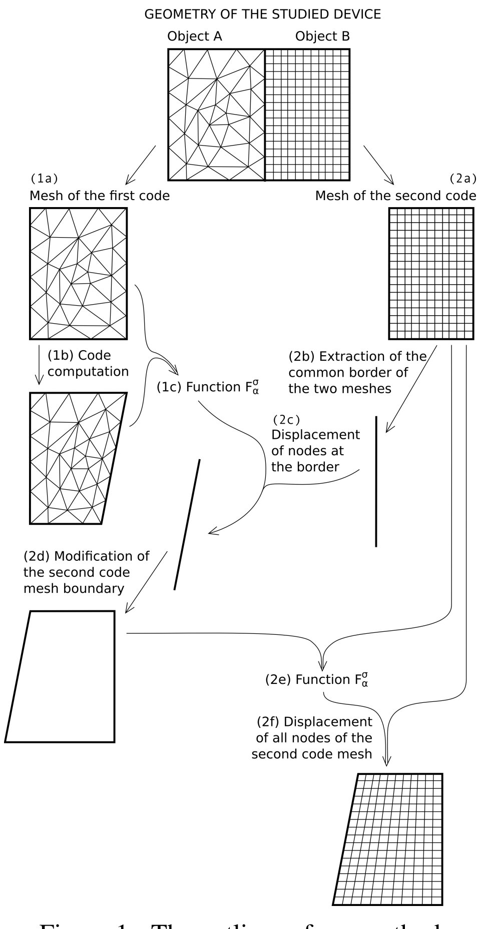
GEOMETRY OF THE STUDIED DEVICE
Object A
Object B
(1a)
Mesh of the first code
(2a)
Mesh of the second code
(1b) Code computation
(1c) Function F_α^σ
(2b) Extraction of the common border of the two meshes
(2c) Displacement of nodes at the border
(2d) Modification of the second code mesh boundary
(2e) Function F_α^σ
(2f) Displacement of all nodes of the second code mesh

Figure 1: The outlines of our method

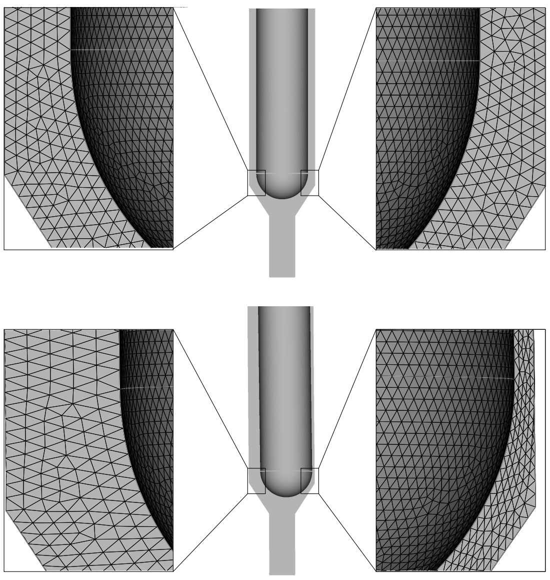

Figure 2: The thermohydraulic code mesh deformation (top: original, bottom: modified by our method)

is launched ($1b$). A function F_α^σ is computed after this step to interpolate the geometrical deformations computed by the mechanical code ($1c$). The initial mesh of the second code is then computed ($2a$), again from the geometry of the whole device. To be able to modify this mesh, its borders are extracted ($2b$). Common parts between these borders and the first code mesh are extracted and displaced ($2c$) applying function F_α^σ previously obtained. These modifications can then be applied to all borders of the second code mesh ($2d$). The boundary of the second code mesh have been displaced. A function F_α^σ is finally calculated to interpolate the evolution of the second mesh border position ($2e$). Once applied to all inner nodes, the second mesh is completely modified ($2f$).

4 EXPERIMENTAL RESULTS

The use case is an experimental device composed by two pipes placed inside a research reactor. The first one, called pressure tube, encapsulates an experiment and isolates it from the rest of the reactor. It is positioned into a second one, called a device holder. Between these pipes, a water flow evacuates the heat emitted by the experiment. The case presented is the thermohydraulic study of the water.

The reactor core heats both pipes leading to anisotropic tube dilatations and water thickness variation. Cooling is no longer homogeneous inducing different pipe deformations and some thermohydraulic phenomena can occur. A coupling between a mechanical code and a thermohydraulic code is valuable. The presented method allows to modify the thermohydraulic code mesh according to the geometrical modifications calculated by the mechanical code. Both initial meshes do not discretise the same geometry. The first one is a mesh of both tubes, whereas the other represent the water separating them. Both meshes share common borders. The first mesh is composed of $14,110$ nodes and $37,330$ tetrahedra, and the second one $223,737$ nodes and $1,025,622$ tetrahedra. Figure 2 illustrates our results. Both tubes are deformed and the water section is then modified. The most important area is located at the bottom of the pressure tube. Pipe deformations involves a water section variation: a shrinkage on one side and an enlargment on the other side. One can see on figure 2 that our method allows to shrink and to stretch smoothly the mesh cells. Computing time of steps is detailed in table 1. The device holder deformation is interpolated by function F_{∞}^{100}, noted $F_{(DH)}$, where $\alpha = \infty$ means the simplification is not processed. The same choice, noted $F_{(PT)}$, is made for the pressure tube. $F_{(W)}$ corresponds to the water mesh modification. Its parameters, chosen experimentally, are $\alpha = 5000$ and $\sigma = 100$. The whole use case calculation is achieved in $13min$ $19s$. In comparison, it is performed in $1h$ $7min$ $37s$ with function F_{α}. The computer used is a PC 64bits under Linux, using a single core at $1.6GHz$ and with $4GB$ of RAM.

	step	computing time	# nodes
$F_{(DH)}$	calculation	2min 04s	9,302
	application	3min 24s	52,453
$F_{(PT)}$	calculation	36s	4,816
	application	1min 22 s	39,962
$F_{(W)}$	calculation	37s	5,000
	application	5min 07 s	131,322

Table 1: Computing time detail for each step

Table 1 shows the method performances in terms of computation time compared with the number of nodes used. For information, the thermohydraulic code requires several days while our method updates its mesh within less than $15min$. As the experimental device study is not yet achieved, we cannot definitely conclude about our method accuracy, but thermohydraulic experts at the CEA validate our results.

5 CONCLUSION AND DISCUSSION

The proposed method allows to transfer deformations computed on a mesh to another mesh sharing common borders. Based on previous works, a new function F_{α}^{σ} is established. It interpolates the displacement of a large number of nodes. This function is the result of a simplification step and the computation of a network of Radial Basis Functions having a compact support. The constraints imposed by the CEA are all well respected. The complete validation of the method will be performed when the device development will be achieved, but according to specialists, results are promising.

An interesting perspective would be to calculate automatically σ, parameter controlling the compact support. For the non-uniform mesh case, one idea might be to establish a grid centered in each RBF center and calculate the value of σ to have all grid cells which contain at least one (or more) other(s) center(s).

ACKNOWLEDGEMENTS

This research is supported by the French Atomic Energy and Energy Alternatives Commission (CEA).

REFERENCES

[BDS07] Bonaccorsi T., Daniel M., Salvo J. D.: Data exchange interface and model for coupling softwares in nuclear reactor simulations. In *Inter. conf. in PLM* (Italy, 2007), pp. 207–216.

[CSW09] Cui T., Song A., Wu J.: Simulation of a mass-spring model for global deformation. *Frontiers of EEE in China 4* (2009), 78–82.

[dBvdSB07] de Boer A., van der Schoot M., Bijl H.: Mesh deformation based on radial basis function interpolation. *Computers & Structures 85*, 11-14 (2007), 784 – 795.

[DGDP10] Duplex B., Gesquière G., Daniel M., Perdu F.: Transfer of mesh deformations between physical codes. In *Curves and Surfaces conference* (june 2010).

[FG08] Frey P. J., George P. L.: *Mesh Generation. Application to Finite Elements.* ISTE, 2008.

[GH97] Garland M., Heckbert P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH* (USA, 1997), pp. 209–216.

[Hel03] Helenbrook B.: Mesh deformation using the biharmonic operator. *Inter. journal for numerical methods in eng. 56*, 7 (2003), 1007–1021.

[OBS05] Ohtake Y., Belyaev A., Seidel H.-P.: 3d scattered data interpolation and approximation with multilevel compactly supported rbfs. *Graphical Models 67*, 3 (2005), 150 – 165.

[PV08] Perdu F., Vandroux S.: System / CFD Coupling for Reactor Transient Analysis. An Application to the Gas Fast Reactor with CATHARE and TRIO_U. *ICANPP* (june 2008).

[RA10] Rendall T., Allen C.: Reduced surface point selection options for efficient mesh deformation using radial basis functions. *Journal of Computational Physics 229* (2010), 2810 – 2820.

[Wen05] Wendland H.: *Scattered Data Approximation.* Cambridge University Press, 2005.

[YK09] Yilmaz A., Kuzuoglu M.: A particle swarm optimization approach for hexahedral mesh smoothing. *International Journal for Numerical Methods in Fluids 60*, 1 (2009), 55–78.