International Forum on Aeroelasticity and Structural Dynamics
IFASD 2019
9-13 June 2019, Savannah, Georgia, USA

A NEW OPEN FLUID-STRUCTURE INTERFACE FOR STEADY AND UNSTEADY NONLINEAR SIMULATIONS

F. G. Di Vincenzo [1]

[1] MSC.Software
4, Rue du Professeur Pierre Vellas
Europarc, Immeuble Jupiter, B10 - 31300 Toulouse, France
faustogill.divincenzo@mscsoftware.com

Keywords: Fluid-Structure Interaction, Highly Flexible Aircraft, FEM, CFD, Morphing, Nonlinear Panel Flutter.

Abstract: A seamless coupling interface called HFSI, has been developed to enable connecting MSC Nastran SOL 400 [1] to any mesh-based CFD solver with all element topologies to allow for efficient and accurate fluid-structure interaction simulation both steady and unsteady. The novel interface enhances a recently developed approach [2] with a new user interface, a novel user defined DMAP module [3, 4], UDMSRV, developed to give a direct access to data from CFD by means of an extended Interface Definition Language (IDL). A new Nastran SubDMAP [3, 4] called SPLINE, that exchanges data with the CFD through the UDMSRV module, has been implemented in the HFSI interface to perform the interpolation at the aero-structure interface and carry out the morphing of the CFD domain. Two different morphing algorithms have been implemented in the SPLINE module. The first one is an enhancement of a method based on the FEM analogy presented in a previous work [2] that has been improved to take care of polyhedral mesh. The second one, here called FEMRIS, is a new hybrid approach developed to combine the fastness of Radial Interpolation Spline (based on Radial Basis Functions), with the accuracy and robustness of the FEM analogy. The latest shows promising results with full scale models where both accuracy and efficiency are important. Sliding and stiffness features have been also added to the morph algorithm to improve the quality and efficiency of the CFD domain deformation. The new coupling approach here presented, where Nastran computes the interpolation at the aero-structure interface and morphing of the CFD domain, really improves the performances of the simulation and robustness of the morphing and thanks to the additional aerodynamic IDL it opens the application to any suited CFD solver. The scope of this work is to illustrate the advantage and benefit of employing such a methodology to real cases and understand the versatility of such an architecture that can be easily extended to any FEM and CFD solvers different from those proposed in the present work. CFD solvers from Cradle [5], that use both tetrahedral and polyhedral mesh, have been chosen for this activity. A transient fluid-structure interaction simulation will be presented: Supersonic nonlinear panel flutter.

1 INTRODUCTION

In a staggered unsteady FSI simulation FEM and CFD solvers run simultaneously and exchange data at each time step and within the time step depending on the coupling strategy [6]. The two codes exchange data at the so called wetted surfaces where the fluid is in “contact” with the solid. Since the mesh discretization at the aero-structure interface differs for the two models an interpolation procedure is indeed needed between the fluid and structure. In most of the standard coupling interfaces available within a FEM solver, as it is the case of OpenFSI, data from and

to CFD cannot be directly retrieved in the interface since the IDL allows to access structural information only and thus it does not provide neither an aero-structure interpolation algorithm nor a tool that computes the morphing of the CFD domain. As a consequence of that the interpolation between the fluid and structure and the morphing must be implemented separately, making the coupling not painless, and the powerful capabilities of the Nastran solver are not exploited for the matrix operations required by the interpolation and morphing. As the number of DOF increases, as it is the case of full scale high-fidelity models, those kind of applications are highly computational expensive. This aspect, if not properly considered in the coupling strategy, can influence the efficiency of the overall process. On the CFD side, one of the most important technical challenge in a fluid-structure interaction simulation that can badly affect the accuracy of the results, if not overcome and properly treated, is the robustness of the morphing strategy. If the CFD mesh does not deform properly around flexible or moving components, the quality of the grid can be easily lost and when negative volumes appear the simulation fails. Furthermore, while ensuring the quality of the finite volumes through the morphing, the procedure has to be highly efficient especially when applied to unsteady simulations where the morphing is carried out at each time step. Another important aspect that pushed this work was the lack of an open fluid-structure interface, as less dependent as possible on the FEM and CFD software, could be easily adapted to different solvers. Here the need to develop an open seamless coupling interface to perform accurate FSI simulations, with an efficient aero-structure interpolation and robust CFD morphing procedure directly embedded and accessible, that enables a straightforward connection of the structural solver to any mesh-based CFD code, where Nastran is extensively exploited to perform all the matrix operations required by the interpolation and morphing to improve the performances of the overall process.

2 PROPOSED APPROACH

The proposed HFSI architecture provides an open fluid-structure interface between MSC Nastran SOL 400 and any CFD solver. A lately developed approach [2] called HSA.OpenFSI to perform steady and unsteady fluid-structure interaction simulations has been redesigned in order to eliminate most of the yet mentioned restrictions and be more flexible, robust and efficient.

The novel interface removes the software pre-requisite of the HSA.OpenFSI strategy that obligates the use of CFD tetrahedral mesh, like in the case of the SCTetra solver, in order to make the employment of the proposed coupling interface open to an enlarged community. The new coupling interface handles any kind of CFD mesh and supports solvers that use polyhedrons, like the SCFlow code that has been employed in this work. The morphing algorithms have been enriched with new methods and functionalities developed to provide additional control on the morphing and be more accurate and efficient with high-fidelity full scale models. The architecture of the interface has been entirely reviewed and redesigned in order to make the load/displacement interpolation procedure between the structure and fluid and morphing of the CFD domain completely independent from the application and run as an additional standalone Nastran solution could be used with any external FEM and CFD solver.

To understand the limits of the coupling strategy proposed by the HSA.OpenFSI and how the novel HFSI architecture proposed in this paper has been developed to be more open and efficient than the previous one, it is important to look at some technical aspects of the IDL of the OpenFSI service the HSA.OpenFSI coupling strategy is based on [7].

2.1 OpenFSI

The OpenFSI functionality provides a mechanism to exchange fluid structure interaction data between the MSC Nastran solver and an external code. The IDL of the OpenFSI service enables to put and get quantities on the structural model only. Since the structural code expects forces at the structural grid points, a "mapping" between the fluid mesh and the structural (wetted) mesh is typically required (the same for the displacements) and needs to be implemented by the user as part of the method. The interface does not provide any IDL that takes care of the interpolation.
Looking more in details at the OpenFSI IDL, the method getWettedNodeForces is called by the structural solver to get wetted node force from external solver, including node force and moment while the method putWettedNodeDispVeloAcce to send wetted node data to fluid solver including displacement, velocity and acceleration, Fig 1.

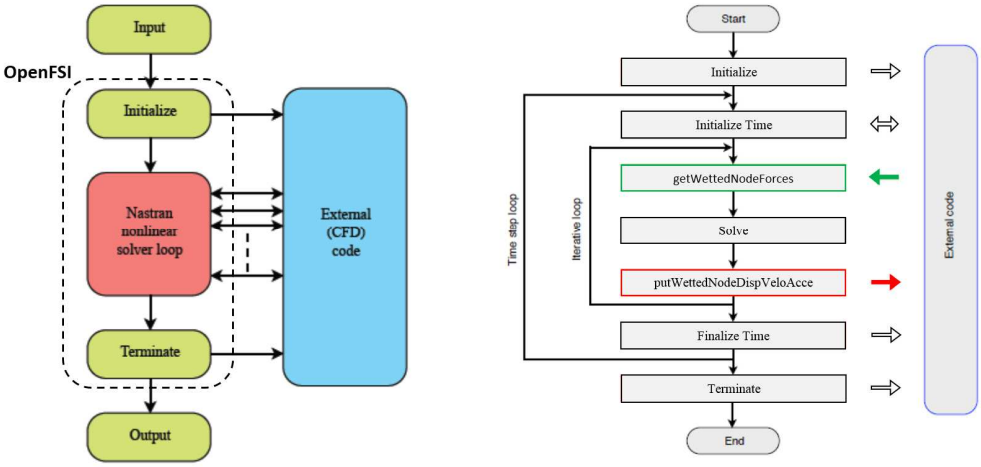


Figure 1: OpenFSI interface

The arguments of these methods are specific structures called wettedNodeEx and they consist in a sequence of a structural grid point ID and six real numbers that define the structural quantities (seqForce, seqDisp, seqVelo and seqAcce) can be exchanged, Fig. 2.

```
struct wettedNodeEx {
    SCAInt64 id;
    SCAReal64 x;    //transitional
    SCAReal64 y;
    SCAReal64 z;
    SCAReal64 rx;   //rotational
    SCAReal64 ry;
    SCAReal64 rz;
};
```

Figure 2: wettedNodeEx sequence

The six real components can be passed to the wettedNodeEx sequence are displacement, including rotations, velocity, including angular velocity, acceleration, including angular acceleration and forces, including moments.
Those quantities cannot be directly exchanged with an external CFD solver in the case the two models use a different discretization, if not properly interpolated on the aerodynamic mesh.

In the HSA.OpenFSI interface an interpolation procedure between the structural and aero grid had been developed based on the spline technology [8] and implemented directly within the service, Fig 3. Such an architecture, as it will be explained, represents a technical limitation and prevents the interface to be easily extended and applied to solvers different from those supported.

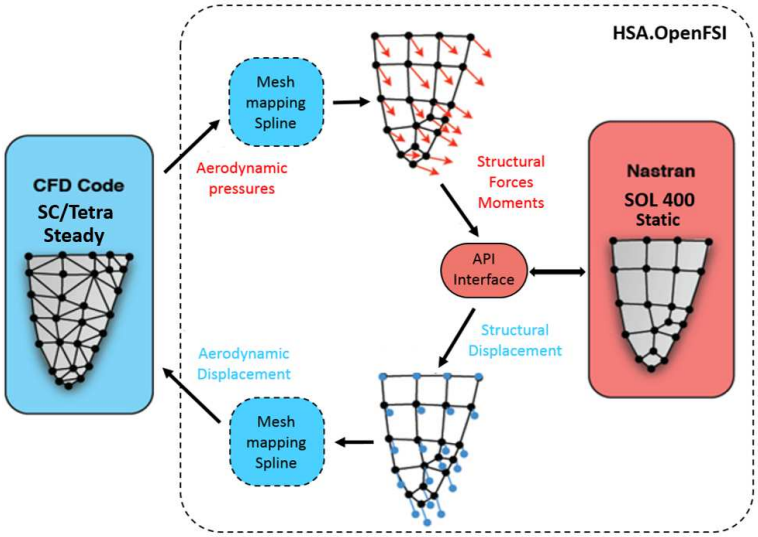


Figure 3: Data exchange between FEM and CFD in HSA.OpenFSI

As a matter of fact, even though the coupling strategy proposed by the HSA.OpenFSI enhances the coupling capabilities thanks to the implementation of the interpolation and morphing procedure which is completely missing in the standard OpenFSI, it cannot be exploited with different CFD solvers and does not get benefit of the powerful capabilities of the Nastran solver to perform the mesh mapping to further improve the efficiency and accuracy.

As described, Fig 3, due to the architecture of the HSA.OpenFSI service the matrix operations required to interpolate data between the two models are performed within OpenFSI by the language chosen to implement the service (C++) rather than by the Nastran solver. On top of that, the interface does support only the two solvers it has been developed to connect to, making the application closed to other codes.

To open the interface to diverse external solvers, instead of implementing a specific OpenFSI service for any suited solver that would require at each time a huge effort of implementation to adapt and maintain the interface over different versions and more over when the codes change, the idea behind the present development was to redesign the interface by getting the interpolation and morphing procedure out of the OpenFSI service and integrating it in a customized and independent SubDMAP Nastran solution called SPLINE, that runs parallel to the SOL 400 and CFD solver and provides its own IDL to get a direct access to both structural and aerodynamic models.
The new coupling strategy proposed by the HFSI interface changes then completely the architecture from that one of the HSA.OpenFSI going from two applications that run simultaneously, the structural and aerodynamic solver, Fig 3, to three applications that proceed at the same time, Fig 4.

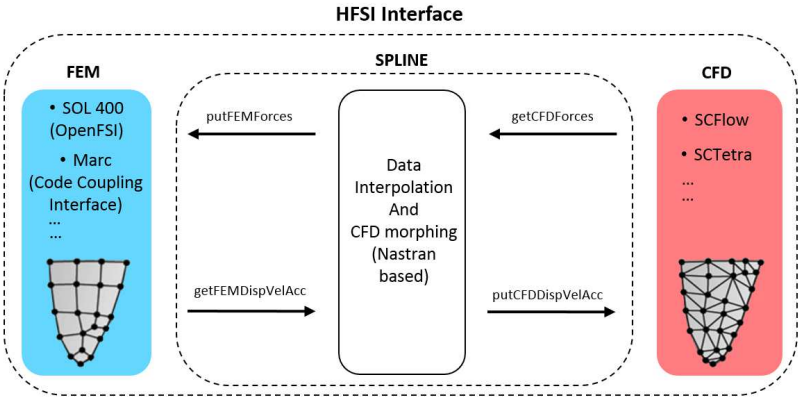


Figure 4: Data exchange between FEM and CFD in HFSI Interface

To make such an architecture it has been necessary to provide specific methods and structures with the IDL of the SPLINE module to enable the communication with CFD and OpenFSI. To accomplish this task it has been required the aid of a user defined module developed within the SPLINE solution, UDMSRV, where the additional IDL has been implemented.
It is now evident how the new architecture could be extended to any FEM and CFD solver thanks to independence and accessibility of the SPLINE module and get benefit of the Nastran capability to perform both aero-structure interpolation and morphing of the CFD domain. The developed interface enables any combination between two different FEM, Nastran and Marc, and two CFD, SCTetra and SCFlow, Fig 4. In the case of the Marc solver, not presented in this paper, the Code Coupling Interface has been employed in place of OpenFSI. The present work will only focus on the coupling interface between the Nastran solver and CFD from Cradle

2.2 HFSI Interface

The aim of this work was that one of building a new open coupling architecture that enables the use of different FEM and CFD solvers while keeping the same aero-structure interpolation and morphing technique [2] based on Nastran. A novel seamless coupling interface called HFSI opens the coupling to any mesh-based CFD solver thanks to an enhanced user interface and IDL that makes the coupling implementation more straightforward and efficient, Fig 5.

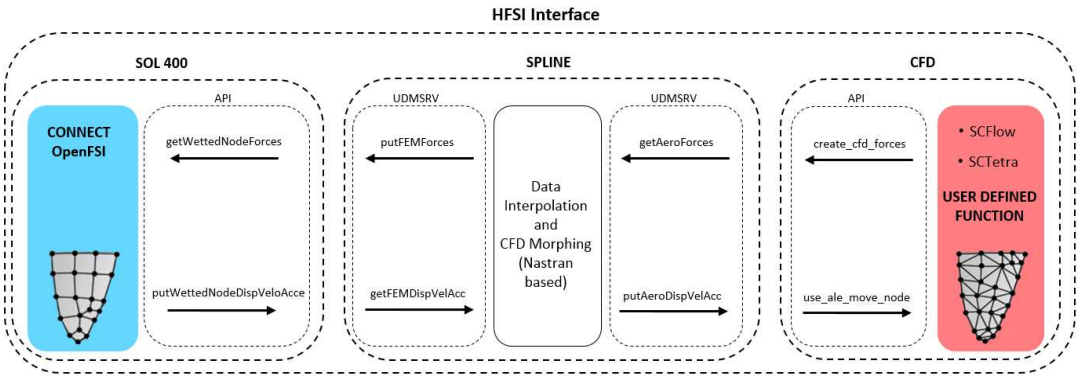


Figure 5: HFSI Interface

The novel interface here proposed presents three separated blocks. The FEM on the left side that computes for the structure, the CFD on the right side that calculates the fluid, and the SPLINE module in the middle that takes care of the interpolation between the two physics and performs the morphing of the CFD domain.

In order to allow both FEM and CFD solvers to communicate to the SPLINE module and exchange model data through it during the coupling, a new IDL with specific methods and structures for both FEM and CFD have been implemented in the SPLINE module by means of a user defined module. The Nastran solver exchanges data with the SPLINE module through OpenFSI while the CFD solver does it through User Defined Functions.

2.3 SPLINE module

As already explained, in the proposed approach the interpolation between the aero and structural mesh and morphing of the CFD domain has been put out of the OpenFSI service and implemented in the SPLINE module. The SubDMAP SPLINE operates on matrix datablocks and runs independently. Here the need to implement an interface within the SPLINE module to enable the connection with external solvers. The SPLINE interface has been developed with the aid of a UDMSRV module.
Unlike the standard OpenFSI interface that lets the user access only data defined on the structural model, the HFSI interface permits to retrieve and put quantities also on the aerodynamic model through the SPLINE module. Four main methods have been implemented in the UDMSRV to send and get quantities to and from the SPLINE module, two for the structural wetted mesh and two other for the aero wetted mesh, Fig 6.

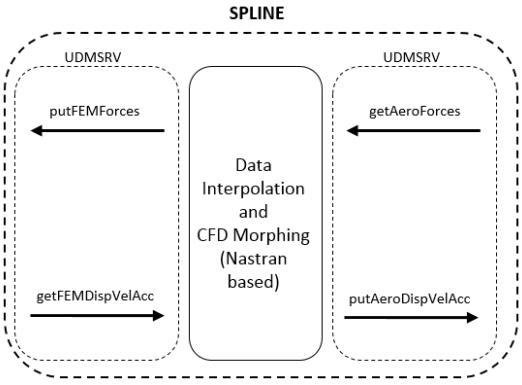


Figure 6: SPLINE module

The functions getAeroForces and putAeroDispVelAcc, used for the communication with the CFD solver, are called to get the aerodynamic nodal forces and put the aerodynamic nodal positions (velocity, acceleration) respectively. The functions getFEMDIspVelAcc and putFEMForces are employed to get the structural nodal positions (velocity, acceleration) and put the structural nodal forces (moment) respectively, to share data with OpenFSI.
The UDMSRV is a straight bridge between the CFD solver, OpenFSI and the SPLINE module. During the simulation the UDMSRV is called at each iteration to pack CFD and FEM quantities into matrix datablocks to allow the SPLINE solution to perform the interpolation (based on the spline technology and HSA Toolkit [8–13]) and morphing. On the other hand, the UDMSRV is asked to unpack the matrix datablocks computed by the SPLINE module and send their content the external solvers, Fig 7 and Fig 8.

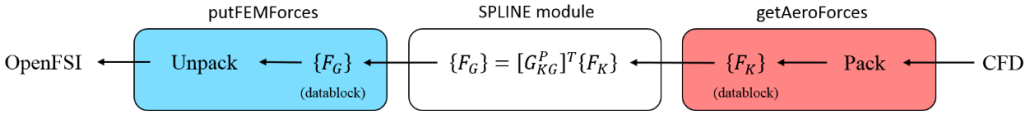


Figure 7: SPLINE module – load interpolation

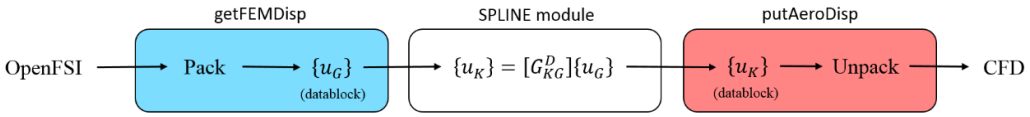


Figure 8: SPLINE module – displacement interpolation

In Fig 7 and Fig 8, $\{F_G\}, \{u_G\}, \{F_K\}$, and $\{u_K\}$ are respectively the force and displacement vectors of the structural wetted nodes and force and displacement vectors of the CFD wetted nodes. The transformation spline displacement matrix, $[G_{KG}^D]$, is employed to transform displacement from the structural grid (G-set) to the aerodynamic grid (K-set), Eq. (1), while the transformation spline load matrix, $[G_{KG}^P]^T$, to transform forces from the aerodynamic grid to the structural grid, Eq. (2).

$$\{u_K\} = [G_{KG}^D]\{u_G\} \tag{1}$$

$$\{F_G\} = [G_{KG}^P]^T\{F_K\} \tag{2}$$

It is now important to understand how the user defined module UDMSRV works, how it is developed to accomplish the functions it is asked to and how it is called by the SPLINE module during the fluid-structure interaction simulation.

2.3.1 UDMSRV User defined module

UDMSRV is a DMAP module [3, 4] that allows users to write their own module functionality. From a DMAP perspective, UDMSRV has the same characteristics as any other DMAP module in that it presents an input datablock list, an output datablock list, a scratch datablock list and a list of parameters, Fig 9.

```
SCA::SCAResult MySrv1::usrDefinedModule(
   const SCA::SCAInt32Sequence& inputDatablockHandles,
   const SCA::SCAInt32Sequence& outputDatablockHandles,
   const SCA::SCAInt32Sequence& scratchDatablockHandles,
   SCA::SCAInt32Sequence& intParams,
   SCA::SCAReal64Sequence& dpParams,
   SCA::SCAStringSequence& strParams)
```

Figure 9: User defined module structure

The UDMSRV may operate on any datablocks passed to it, modifying the contents of the datablocks and creating new ones; this provides an extremely powerful capability.
The user defined behavior of a UDMSRV module is accomplished by developing a program that carries out a set of operations, possibly that operate on the input and output datablocks and parameters specified on the call to the UDMSRV module. Like OpenFSI [6], this capability is available through the Service Component Architecture or Service Oriented Architecture (SCA) [7]. The SCA Kernel allows a user defined program, written in high level language, to be compiled in the Software Development Kit (SDK) environment into a dynamically linkable object, which appears as a dynamic link library (.dll) or shared object (.so) depending on the operating system.
To better understand how a UDMRSV is called and executed within a Nastran solution sequence a simple example is here presented. For simplicity, a DMAP alter for SOL 100 (any solution sequence can be indeed employed) is written to just call the UDMSRV and then stop.

To connect any Nastran solution sequence to a UDMRSV module the CONNECT Nastran card has to be inserted in the File Management Section of the input file, Fig 10.

```
CONNECT SERVICE HWSRV 'hwmod'
SOL 100
COMPILE USERDMAP LIST
ALTER 2 $
MESSAGE //' Calling dmap udmsrv ' $
UDMSRV //'HWSRV' $
END $
CEND
BEGIN BULK
ENDDATA
```

Figure 10: SOL 100 DMAP alter input file

The UDMSRV, implemented in C++ language for the sample, is just asked to write a text in the f06 file right after the DMAP MESSAGE 'Calling dmap udmsrv'. To accomplish this, the module requires the aid of the printer server function distributed with the SCA capability, Fig 11.

```
SCA::MDSolver::System::PrintServer::SCAIPrintServer printer;
SCA::SCAIService spSrv = m_serviceAccess->getService("SCA.MDSolver.System.PrintServer");

printer->print("=====================================================");
printer->print(" *** In MySrv1 of the User Defined Module Service ***");
printer->print("=====================================================");
```

Figure 11: UDMRSRV – extract of the cpp file

These instructions load and initialize the print server. Once the print server is defined the module prints out the message and gives back the control to Nastran. Running the SOL 100 DMAP alter, Fig 12, the message is found in the f06 file as expected, Fig 12.

```
 ^^^ CALLING DMAP UDMSRV
=====================================================
 *** In MySrv1 of the User Defined Module Service ***
=====================================================
```

Figure 12: Extract of the f06 file

We can now look at how the SPLINE module exchange data with the CFD solver and OpenFSI through the specific UDMRSV module developed for the new coupling architecture. An extract of the SubDMAP SPLINE that performs the load interpolation is shown, Fig 13.

```
$     *******   GET CFD LOAD AND PACK THE MATRIX   *******
UDMSRV ,/ FORCEA/ 'mysrv1' $

$     *******   LOAD INTERPOLATION   *******
MPYAD XGPGK0,FORCEA, / FORCES/$

$     *******   UNPACK THE MATRIX AND SEND THE LOAD TO OPENFSI   *******
UDMSRV  FORCES// 'mysrv1' $
```

Figure 13: SPLINE module – load interpolation

The SPLINE solution calls the UDMRSV to get the load from the CFD solver. The load is packed into a datablock called FORCEA ($\{F_K\}$), and the SPLINE module carries out the load interpolation, Eq. 2, by multiplying the transformation spline load matrix XGPGK0 by the

aerodynamic force vector FORCEA. The computed structural load stored in the datablock FORCES ($\{F_G\}$) is unpacked and sent to OpenFSI through another call to the UDMSRV.

In order to work on datablocks the user defined module requires the aid of the tools known as GINO that stands for General Input Output. The SCA interface provides an API for GINO through the GinoEmb IDL. In the initialization of the UDMSRV the GinoEmb server is loaded to access datablocks. A new datablock object called Datablock_cfdLoad is created and the file handle of the output datablock is associated to it. Once the datablock reserved to the CFD load has been created the function getAeroForces is called to populate the datablock with the aerodynamic load received by the CFD solver, Fig 14.

```
SCA::SCAResult MySrv1::usrDefinedModule(
    const SCA::SCAInt32Sequence& inputDatablockHandles,
    const SCA::SCAInt32Sequence& outputDatablockHandles,
    onst SCA::SCAInt32Sequence& scratchDatablockHandles,
    SCA::SCAInt32Sequence& intParams,
    SCA::SCAReal64Sequence& dpParams,
    SCA::SCAStringSequence& strParams)
{

   /*******    Load and setup GinoEmb service object Datablock_cfdLoad    *******/

   SCA::SCAIService sp_gino = m_serviceAccess->getService("SCA.MDSolver.Io.GinoEmb");
   SCA::MDSolver::Io::GinoEmb::SCAIGinoEmbService Datablock_cfdLoad;

   /*******    Get Gino File Handle for Datablock_cfdLoad    *******/

   SCA::SCAInt32 gfilept1 = outputDatablockHandles[0];

   /*******    Set the file handle to Object    *******/

  Datablock_cfdLoad->setFileHandle( gfilept1 );

   /*******    get and pack a matrix    *******/

  (void) getAeroForces( Datablock_cfdLoad );
```

Figure 14: UDMSRV service – CFD Load datablock object

The argument passed to the function developed in getAeroForces to populate the Datablock_cfdLoad is a structure called wettedAeroNode and it consist in a sequence of an aerodynamic grid point ID and six real numbers that define the aerodynamic quantities can be exchanged, Fig. 15.

```
struct wettedAeroNode {
    int* id;    // aero id
    double* x;
    double* y;
    double* z;
    double* rx;
    double* ry;
    double* rz;
};
```

Figure 15: wettedAeroNode sequence

The real components can be aerodynamic node position, displacement, velocity, acceleration and forces. The same structure is used by the function putAeroDispVelAcc to send data to the CFD solver.

Once the SPLINE module has performed the load interpolation the UDMRSV is called to unpack the datablock FORCES that contains the structural load and send its content to OpenFSI, Fig 16.

```
SCA::SCAResult MySrv1::usrDefinedModule(
    const SCA::SCAInt32Sequence& inputDatablockHandles,
    const SCA::SCAInt32Sequence& outputDatablockHandles,
    onst SCA::SCAInt32Sequence& scratchDatablockHandles,
    SCA::SCAInt32Sequence& intParams,
    SCA::SCAReal64Sequence& dpParams,
    SCA::SCAStringSequence& strParams)
{

   /*******    Load and setup GinoEmb service object Datablock_femLoad    *******/

   SCA::SCAIService sp_gino = m_serviceAccess->getService("SCA.MDSolver.Io.GinoEmb");
   SCA::MDSolver::Io::GinoEmb::SCAIGinoEmbService Datablock_femLoad;

   /*******    Get Gino File Handle for Datablock_femLoad    *******/

   SCA::SCAInt32 gfilept1 = inputDatablockHandles[0];

   /*******    Set the file handle to Object    *******/

  Datablock_femLoad->setFileHandle( gfilept1 );

   /*******    unpack a matrix and send    *******/

  (void) putFEMForces( Datablock_femLoad );
```

Figure 16: UDMSRV service – FEM Load datablock object

The inputDatablockHandles list is used to get the datablock of the structural load computed by the SPLINE module as an input. The function putFEMForces unpack the datablock Datablok_femLoad and sent its content to OpenFSI. To maintain the same terminology and facilitate the exchange with the structural solver a structure equivalent to the wettedNodeEx, Fig 2, provided by OpenFSI has been implemented in the UDMSRV module and used to send the load and receive the displacement from and to OpenFSI.

As far as the displacement interpolation is concerned the SPLINE solution calls the UDMRSV to get the displacement from OpenFSI via the function getFEMDisp. The displacement is packed into a datablock called DISSTR ($\{u_G\}$), and the SPLINE module carries out the interpolation, Eq. 1, by multiplying the transformation spline displacement matrix $[T]$, that is the transpose of XGPGK0 by the structural displacement DISSTR. The computed aerodynamic displacement stored in the datablock AEDISP ($\{u_K\}$) is unpacked and sent to the CFD solver through the UDMSRV via the function putAeroDispVelAcc, Fig 17.

```
$     *******   GET STRUCTURAL DISP AND PACK THE MATRIX   *******
UDMSRV ,/ DISSTR/ 'mysrv1' $

$     *******   DISP INTERPOLATION   *******
MPYAD T,DISSTR, / AEDISP/$

$     *******   UNPACK THE MATRIX AND SEND DISP TO CFD   *******
UDMSRV  AEDISP// 'mysrv1' $
```

Figure 17: SPLINE module – displacement interpolation

The development of the interface implemented within the UDMSRV module required additional development of the IDL to exchange CFD data to perform the CFD morphing.

The morphing algorithms that have been developed in the new coupling interface are described in the next sections.

2.4 CFD Mesh morphing

As already mentioned, two different morphing algorithms have been implemented in the proposed interface. The first one is an enhancement of what was lately presented in [2], based on the FEM analogy, to support polyhedral mesh. Even though this method demonstrates to be accurate and robust it is not really efficient when applied to models with a huge number of DOF. The second one is a novel hybrid approach that combines the powerful capabilities of the FEM analogy with the efficiency of the Radial Interpolation Spline available in Nastran [8]. The latest has been specifically developed to improve the performances of the first approach while keeping the same accuracy and robustness. On top of the stiffness and sliding capabilities already presented in [2], some new functionalities have been developed to improve the efficiency and quality of the morphing. One new feature enables boundary layers around deformable bodies to deform freely without any constraint that comes from the rest of the domain. Another feature permits the definition of morphing regions, where the internal nodes can move independently and nodes on the boundary can slide on it, and not-morphing subdomains that share interfaces with the morphing regions where the fluid quantities are exchanged. The latest feature enhances the efficiency of the morphing since it operates only a reduced region of the computational domain and in the same time improves the quality of the morphing because it removes the fixed constraints at the boundary.

2.4.1 FEM analogy

The SCFlow CFD solver employed for this activity uses arbitrary polyhedrons. An example of a polyhedral mesh of the uCRM [14] is presented, Fig 18.

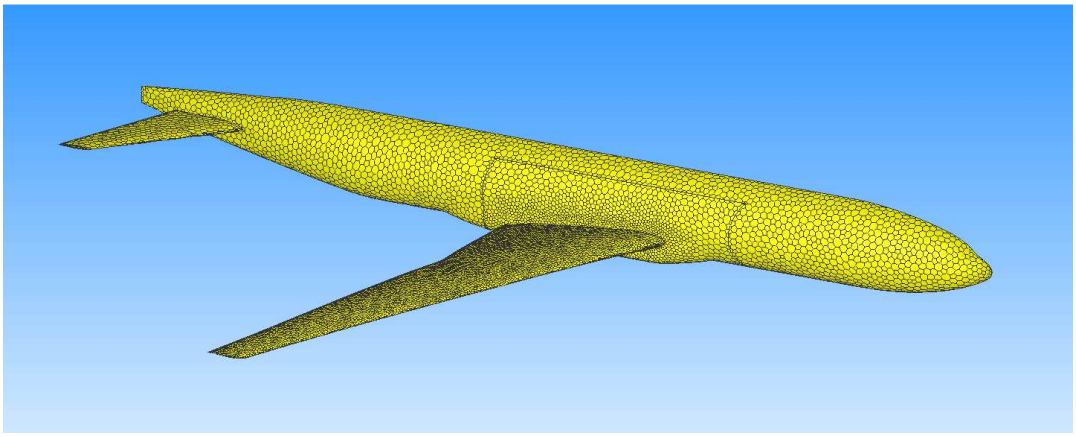


Fig. 18: A general polyhedral mesh – uCRM model

The morphing algorithm based on the FEM analogy developed in [2] has been extended to handle polyhedral mesh and improved to add more control on the morphing of the boundary layer. In the initialization phase of the coupling the computational domain or a subdomain of it that encapsulates the deformable walls is transformed into an equivalent linear Nastran FEM model called FEM_{CFD}. CBEAM elements are used to connect the nodes that constitute the edge of the face of a polyhedron. Material and element properties are defined in order to have less deformation in the area close to the deformable surfaces and more as the elements move far from the deformable wall, Fig 19.

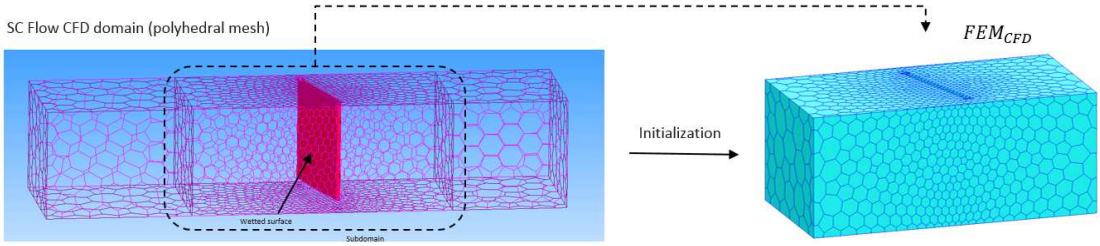


Fig. 19: Flap in a duct. Polyhedral mesh and subdomain FEM_{CFD} model

To maintain the shape of the computational domain, boundary conditions are imposed on the nodes that lie on the boundaries. To improve the quality of the morphing while keeping the shape of the computational domain unchanged, in the case that the boundary surfaces of the selected subdomain are plane, the sliding functionality could be employed to constrain to zero the out-of-plane displacement and allow the nodes to move on the plane (SPC1 cards). Enforced displacement conditions $\{Y_S^D\}$ are applied on the nodes of the wetted surface (combination of SPCD and SPC1 cards), Fig. 20. The displacement enforced on the wetted surface are those ones computed by Eq. 1.

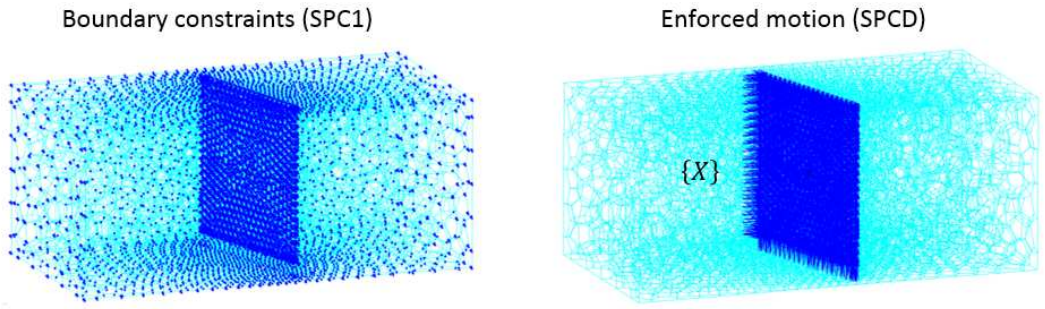
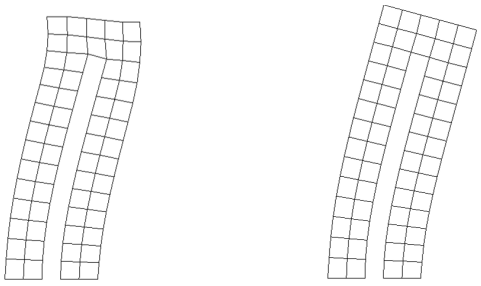


Fig. 20: Flap in a duct. Boundary conditions and enforced motion

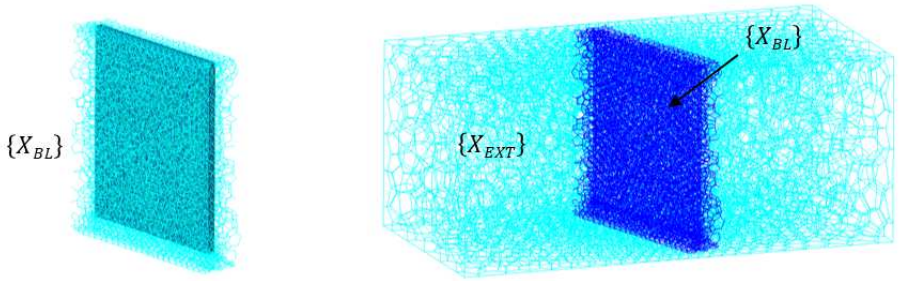
Boundary conditions in some cases negatively affect the morphing accuracy and lead to undesired deformation of the elements in the area close to the flexible body (boundary effect), Fig 21 (a). The morphing algorithm has been modified in order to allow the boundary layer, or a user defined number of layers around the flexible surfaces, where the accuracy of the solution is rather really important, to deform freely without considering the effect of the applied boundary conditions, Fig 21 (b).



(a) (b)

Fig. 21: Mesh morphing – boundary effect

The FEM_{CFD} model is partitioned into two subdomains, the boundary layer model $FEM_{CFD_{BL}}$, Fig 22 (a) that corresponds to the computational domain given by the layers around the flexible surface and includes the wetted surface, and the rest of the domain $FEM_{CFD_{EXT}}$, Fig 22 (b).



(a) (b)

Fig. 22: $FEM_{CFD_{BL}}$ and $FEM_{CFD_{EXT}}$

A DMAP alter solution [3] of a SOL 101 is performed on the FEM_{CFD} within the SPLINE module to extract the matrix $[K_{fs}]$ and $[LLL]$ of each subdomain needed to compute the solution in the morphing domain. The matrix $[LLL]$ is the sparse lower triangular factor/diagonal reduced from $[K_{LL}]$ that is the stiffness matrix of the FEM_{CFD} model. The matrix $[K_{fs}]$ is the partitioned stiffness matrix that allows to reduce the static load vector $\{P_f\}$ on free nodes given an enforced displacement vector $\{Y_S\}$, Eq. 3.

$$\{P_f\} = -[K_{fs}]\{Y_S\} \tag{3}$$

The linear system to be solved, Eq. 4, is then decomposed into two sequential linear problems and the solution $\{X\}$ of the morphing computational domain is given by the solution of the boundary layer, $\{X_{BL}\}$, Eq. 5, and the solution of rest of the model $\{X_{EXT}\}$, Eq. 6.

$$[K_{LL}]\{X\} = \{P_f\} \tag{4}$$

$$[K_{LL_{BL}}]\{X_{BL}\} = \{P_{f_{BL}}\} \tag{5}$$

$$[K_{LL_{EXT}}]\{X_{EXT}\} = \{P_{f_{EXT}}\} \tag{6}$$

The linear system that computes the grid displacement $\{X_{BL}\}$ of the boundary layer $FEM_{CFD_{BL}}$, Eq. 5, can be rewritten as Eq. 7:

$$[LLL_{BL}][D_{BL}][LLL_{BL}]^T\{X_{BL}\} = \{P_{f_{BL}}\} \tag{7}$$

In the previous equation $[D_{BL}]$ and $[LLL_{BL}]^T$ are respectively the diagonal matrix and the upper triangular factor matrix from $[K_{LL_{BL}}]$ and $\{P_{fbl}\}$, that are stiffness matrix of the $FEM_{CFD_{BL}}$ model and reduced load vector on free nodes, Eq. 8.

$$\{P_{f_{BL}}\} = -[K_{fs_{BL}}]\{Y_{BL}\} \tag{8}$$

The enforced displacement vector $\{Y_{BL}\}$ contains the aerodynamic displacement $\{u_K\}$ computed on the wetted surface during the fluid-structure interaction simulation by Eq. 1 and the DOF of

constrained nodes on the external boundaries within the boundary layer. The solution $\{X_{BL}\}$, Eq. 7, is computed through a Forward-Backward Substitution (FBS) as presented in [2].
Once the solution X_{BL} has been calculated, the nodal displacements of the external layer of the boundary layer are used as an enforced motion, $Y_{BL_{EXT}}$, to morph the subdomain $FEM_{CFD_{EXT}}$, where $\{P_{f_{EXT}}\}$ is now given by Eq. 9, and solution is calculated in all the computational domain.

$$\{P_{f_{EXT}}\} = -[K_{fs_{EXT}}]\{Y_{BL_{EXT}}\} \tag{9}$$

The sliding boundary conditions is a really powerful capability especially in applications where small gaps are present, Fig 23 (only the polyhedrons that have faces on the boundary and wetted surface are shown).

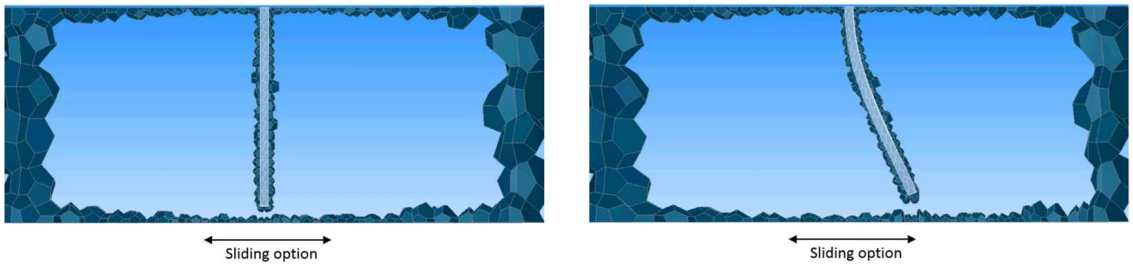


Fig. 23: Mesh morphing with sliding conditions

Without such a feature, the elements within the gap would stretch too much and the mesh would not pass the check quality performed by the solver and the simulation would fail. This functionality prevent also to resort to the re-meshing and overset mesh capabilities that even though are really helpful in these kind of applications, they are still highly computational expensive.
As previously mentioned and demonstrate [2], the morphing algorithm based on the FEM analogy is really accurate and robust. On the other hand the efficiency needs to be improved when the number of DOF is really high. The time required by the solver to extract the matrix $[K_{fs}]$ and $[LLL]$ needed by the approach to solve the morphing of the computational domain can is some cases take too long.
For this reason, with the aim of improving the performances of the method while keeping its accuracy, a new algorithm called FEMRIS has been implemented in the SPLINE module that combines the FEM analogy with the efficiency of the Radial Interpolation Spline. The novel method will be explained in the next section.

2.4.2 FEMRIS

The FEMRIS morphing algorithm combines the FEM technology described in the previous section with the Radial Interpolation Spline (RIS) available in Nastran. The RIS technique is available through the SPLINE 4 technology [8], Fig 24.

```
SPLINE4 1       999999991               1               RIS     BOTH
                        WF2     4.854887
AELIST  1       100545  100546  100547  100548  100549  100550  100551
        100552  100553  100554  100555  100556  100557  100558  100559
        100560  100561  100562  100563  100564  100565  100566  100567
        ..
        ..
SET1    1       1       2       18      27      35      44      49
        69      78      86      114068  114069  114070  114071  114072
        114073  100546  100878  114074  114075
```

Fig. 24: SPLINE 4 definition – RIS

SPLINE 4 is a curved surface spline can be employed for interpolating motion or forces on general aerodynamic geometries. Its definition is quite simple. It needs a list of points to be to be morphed, AELIST, that in our application are the aerodynamic nodes of the computational fluid domain and a list of control points, SET1, used as masters to compute the morphing given their displacement. The control points are positioned on the deformable surface, to account for the movement of the wetted surface, and boundaries of the morphing domain, to preserve the geometry of the computational domain. Since the time required by this technique increases with respect of the number of control points used it is really important to limitate the number of master nodes selected. The RIS technique is technology based on the Radial Basis Functions (RBF). Two different Wendland interpolation functions are available with the RIS [8], WF1 in Eq. 10 and WF2 in Eq. 11.

$$\phi\left(\frac{r}{r_c}\right) = \left(1 - \frac{r}{r_c}\right)_t^2 \tag{10}$$

$$\phi\left(\frac{r}{r_c}\right) = \left(1 - \frac{r}{r_c}\right)_t^4 \left(4\frac{r}{r_c} + 1\right) \tag{11}$$

Where:

$$(y)_t = \begin{cases} y & if\ y > 0 \\ 0 & if\ y < 0 \end{cases} \tag{12}$$

In Eq. 10 and Eq. 11 r_c is the radius of support of radial interpolation function and r is the distance from the control point.

To show how this approach works, the FEMRIS technique is applied to the morphing of a simple 2D mesh with 13523 nodes and 13238 quadrilateral elements. We can consider as if a deformable body was surrounded by a computational fluid domain. All the nodes of the the wetted surface of the flexible body (blue dots), where an arbitrary shape is imposed, are considered as master nodes. To constraint the boundary of the domain, four corners and two additional points on the bottom of the solid, two on the opposite sides and two others on the lateral boundary (red dots), have been defined as control points with the displacement constrained to zero. Fig 25.

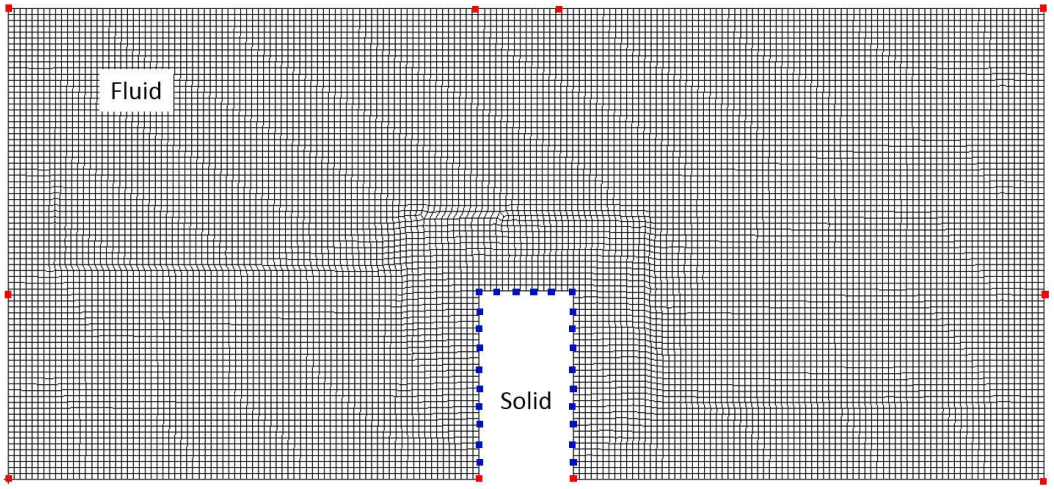


Fig. 25: 2D mesh – Control points for RIS

A first pure RIS simulation is first performed without employing the aid of the FEM analogy and without any sliding condition. Apart from the bottom side where no control points are defined, the rest of the boundaries maintain the shape unchanged, Fig 26. The morphing is properly performed even though the elements around the deformable body deform considerably to satisfy the boundary conditions imposed on the top of the domain where the nodes are not allowed to move.

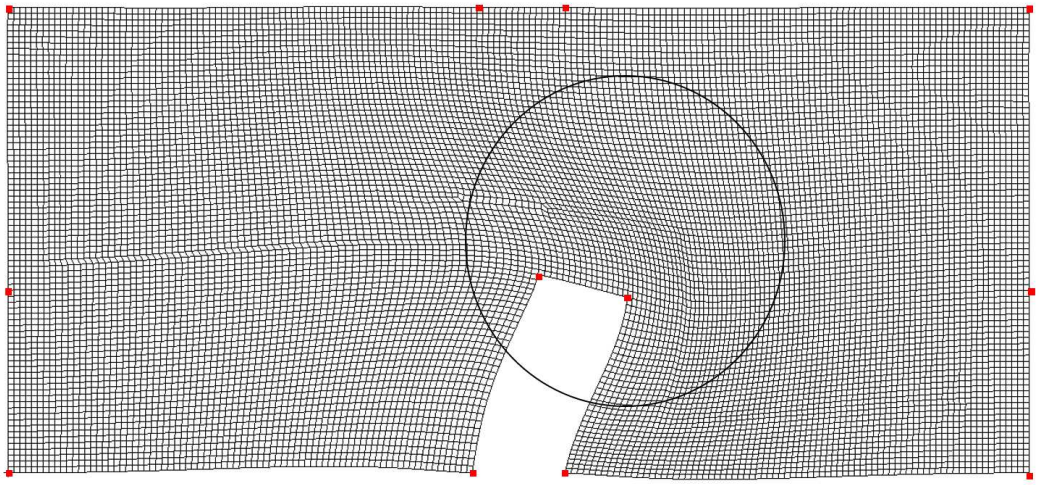
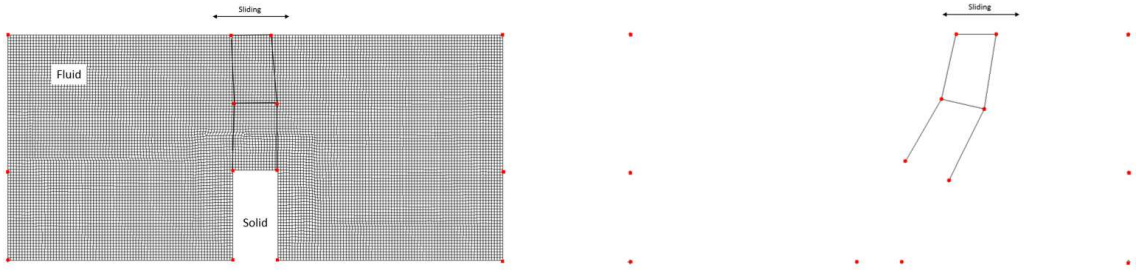


Fig. 26: 2D mesh – RIS Morphing and control points

To take advantage of the FEM analogy and improve the quality of the morphing without decreasing the performances, the technology developed in the FEMRIS employs the aid of a reduced and simplified model $ROFEM_{CFD}$, built behind the scene by the SPLINE module to capture the global behavior of the deformation of the computational domain, given the enforced displacement on the flexible solid, and uses this solution as a master displacement to guide the RIS technology perform the morphing.
For the example here presented, the simplified $ROFEM_{CFD}$ model consists of only 6 structural grids connected by 6 BEAM elements. Two are the upper corners of the solid, two those ones used previously placed on the upper boundary and two others between the two couples. The sliding functionality has been used to allow the structural nodes that lie on the boundary of the fluid domain to slide on the edge, Fig 27 (a).



(a) (b)

Fig. 27: 2D mesh – FEMRIS definition and control points

Before performing the solution on the computation domain, the FEMRIS computes an intermediate solution $X_{ROFEM_{CFD}}$ of the reduced $ROFEM_{CFD}$ model by imposing the displacement of the wetted surface $\{Y_S\}$ of the solid, Eq. 13 and Eq. 14.

$$\left[K_{LL_{ROFEM}}\right]\{X_{ROFEM}\} = \left\{P_{f_{ROFEM}}\right\} \tag{13}$$

$$\left\{P_{f_{ROFEM}}\right\} = -\left[K_{fs_{ROFEM}}\right]\{Y_S\} \tag{14}$$

Once the reduced model has been solved, Fig 27 (b) the obtained displacement $X_{ROFEM_{CFD}}$ is put into the control point displacement vector $\{u_C\}$ along with the control points that serve to preserve the shape of the boundary of the domain as in the previous simulation. The solution of the computational domain in then solved by Eq. 15.

$$\{u_K\} = \left[G_{KC}^{D}\right]\{u_C\} \tag{15}$$

In the previous equation $\left[G_{KC}^{D}\right]$ is the transformation RIS spline displacement matrix that computes the displacement on aerodynamic grids $\{u_K\}$, CFD domain nodes, given the displacement known on the control points $\{u_C\}$.
When the morphing is performed it can be observed as the combination of the RIS method and the FEM analogy, along with the sliding functionality, really helps the deformation of the domain and allows to achieve a smoother morphing. The mesh does not present anymore the distortion of the elements close to solid boundary as it has been found with the pure RIS technique, Fig 28. No boundary layer morphing functionality has been considered. The small distortion found on the lateral sides, as those already mentioned on the bottom, can be removed by imposing additional control points and constrain the displacement to zero.

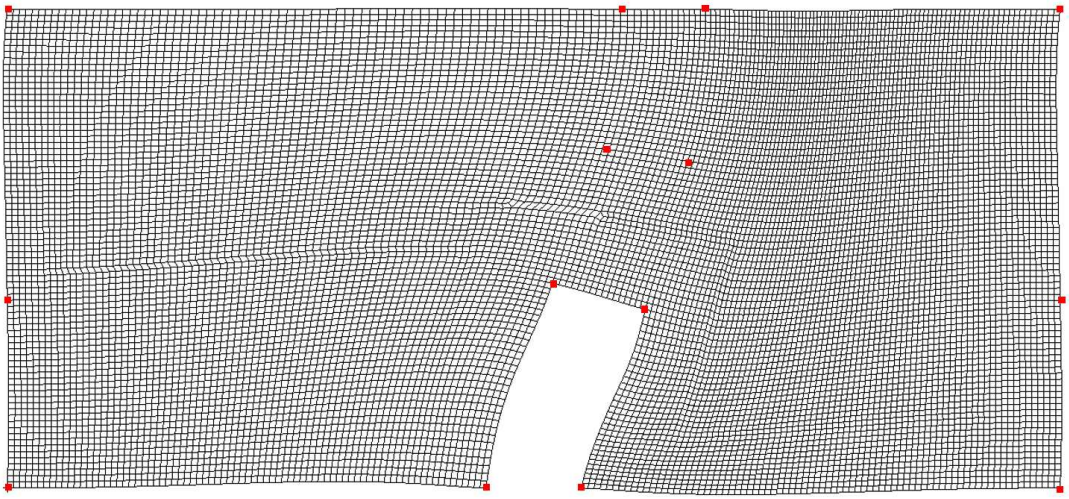


Fig. 28: 2D mesh – FEMRIS Morphing and control points

This technology shows then the advantage of employing a really efficient procedure based on the RIS technique while keeping the accuracy of the FEM analogy and its added functionalities.

One more procedure can be employed to further enhance the performances and quality of the morphing is the definition of interfaces between the morphing domain and non-morphing regions. The SCFlow CFD solver is really suitable for such a strategy.
In order to reduce the computational time required by the morphing algorithm and improve then the efficiency, the computational domain can be partitioned into several subdomains where only that one which encapsulates the deformable body (wing in the example) can deform, Fig 29. To take advantage of the sliding functionality and morph the grid within the morphing region

without the constraint of fixed nodes at the boundaries, interface regions can be defined at the shared surfaces between the moving region, Fig 30 (a), and the static region, Fig 29 (b), where the fluid quantities are interpolated from one mesh, own by the morphing volume, to the other one, own by the non-morphing region.

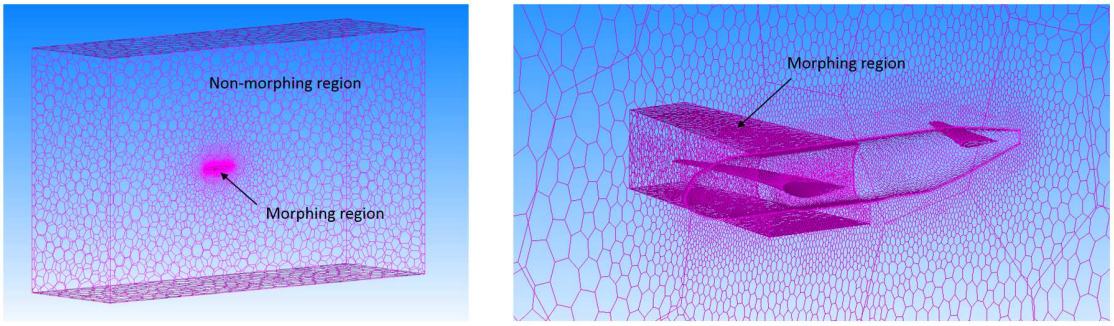
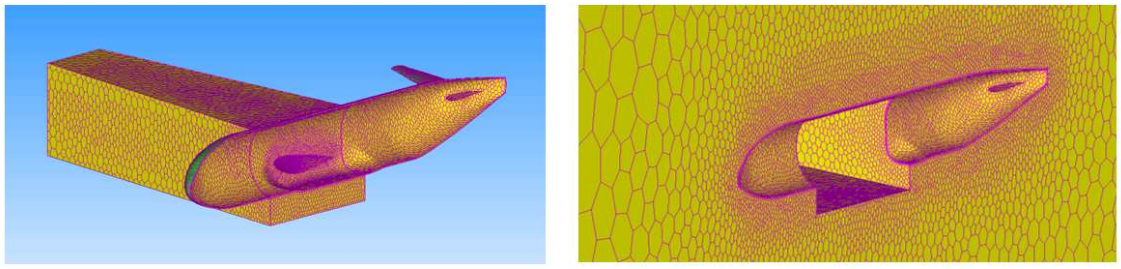


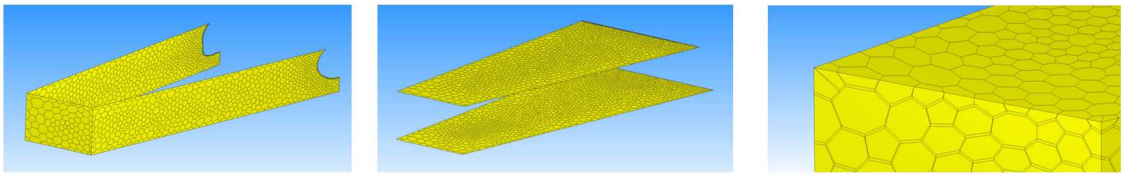
Fig. 29: uCRM – Subdomain definition



(a) (b)

Fig. 30: uCRM – Morphing and non-morphing regions

The CFD solver creates a couple of mesh at any interface. The mesh own by the morphing region is called moving while that own by the non-morphing region is called static. Different boundary conditions can be applied to the interfaces. In the example here presented, sliding conditions have been assigned to the lateral boundary of the morphing region, Fig 31 (a), while the upper and lower surfaces have been defined as fixed, Fig 31 (b). The nodes of the moving mesh own by the morphing region move on the plane while those ones of the static maintain the original position.



(a) (b) (c)

Fig. 31: uCRM – Boundary conditions and mesh interfaces

3 NUMERICAL RESULTS

Analyses to assess the developed interface to perform nonlinear fluid-structure interaction are discussed in Sec. 3.1. The proposed methodology is applied to study the nonlinear panel flutter of a supersonic plate.

3.1 Nonlinear Panel Flutter of a Supersonic Plate

The plate is a rectangular thin flexible structure, long and narrow, clamped on a rigid support, flying at Mach number of 2.4 at sea level. The flow runs parallel to the longest edge of the plate on the top side only. The flexible structure is subjected to both CFD loading and noise due to the turbulent boundary layer modeled through random pressure fluctuation around 1 bar, Fig 32.

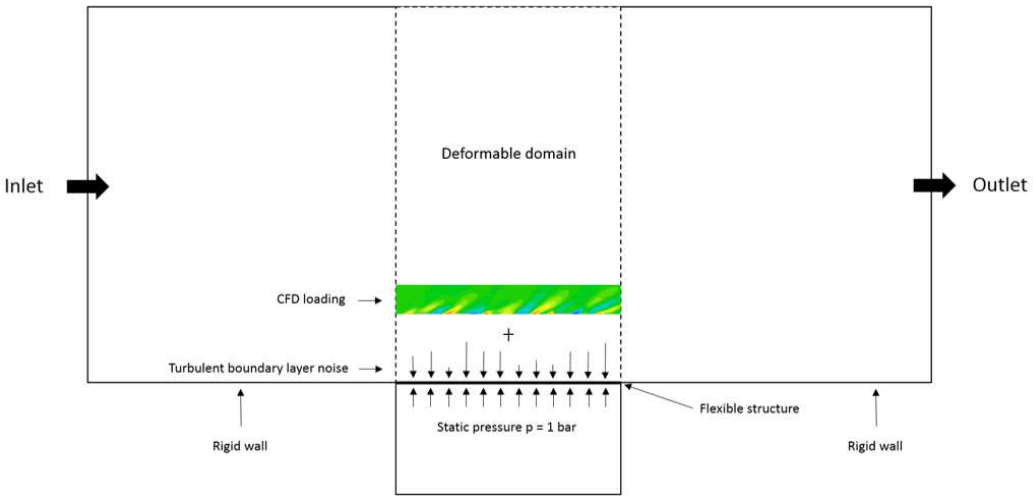


Fig. 32: Aeroelastic model – boundary conditions

Elasto-plastic material properties have been used to characterize the nonlinear behavior of the plate that is made of quadrilateral shell elements. The structural model is constrained at the boundaries, Fig 33, and a static pressure load of 1 bar has been added as a mechanical load that acts on the bottom of the plate to be consistent with the reference pressure used in the CFD calculation.

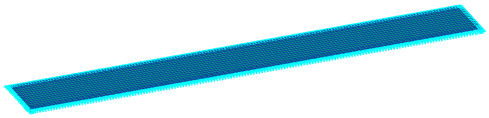


Fig. 33: Plate FEM – Boundary conditions

In order to better capture the shock waves that develop during the simulation a density-based solver has been preferred to a pressure-based one with a RNG k-EPS turbulent model and temperature activated. A dual-time stepping method has been chosen with inner loops within the cycles. The loop strategy is mandatory with a density-based solver to reach the convergence at each iteration even though the computational time required by the solvers increases with the number of loops. The CFD model has 1157281 tetrahedral elements, Fig. 34.

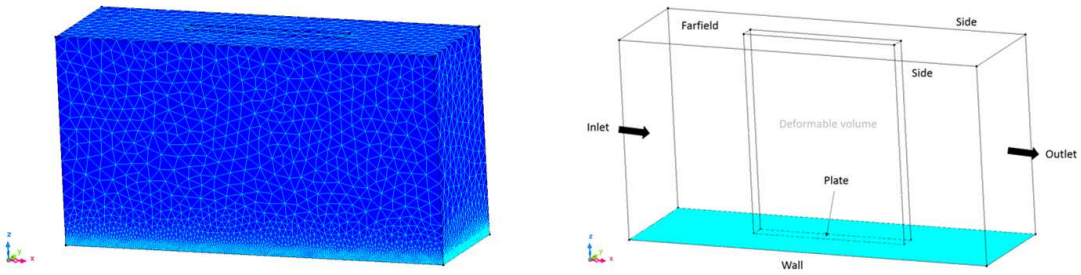


Figure 34: CFD mesh and boundary conditions

Some layers with a much smaller element size have been positioned close to the wall in order to properly predict the flow field in the area where the structure deforms and then be able to well capture the shock and expansion waves that develop during the simulation. The velocity at the inlet is set to 825 m/s in the x direction. Free sleep wall conditions have been chosen for the plate and the rigid support the flexible structure is supported to (wall). A natural inflow/outflow B.C. is defined for the Outlet with a temperature T = 20°. Wall free sleep condition is specified for the Fairfield and Sides.
The computational domain has been partitioned in such a way that only the region called Deformable Volume, which is a rectangular prism where the base corresponds to the Flap and runs till the farfield, is defined as the morphing region and used as FEM_{CFD} (75065 polyhedrons and 15221 nodes). A higher stiffness has been used in the boundary layer (green elements), while a lower value in the area outside the boundary layer (red elements), Fig 35.

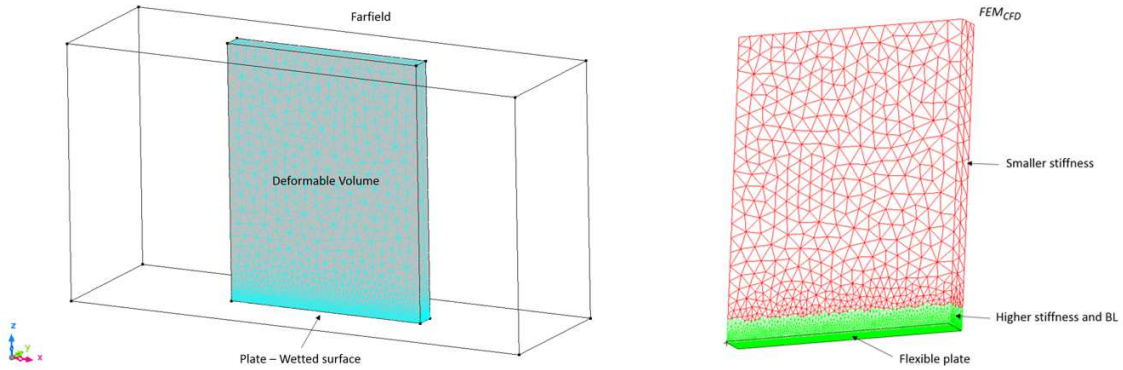


Figure 35: Morphing region – FEM_{CFD} model with Stiffness options

Since the plate is clamped, and rotations blocked, it is not needed the use of interfaces between the morphing and non-morphing regions and the sliding condition have not been employed. The displacement of the nodes of the four lateral faces and top surface of the morphing cuboid have been constrained to zero. The algorithm based on the FEM analogy has been used for this application to perform the morphing of the computational domain, Fig 36.

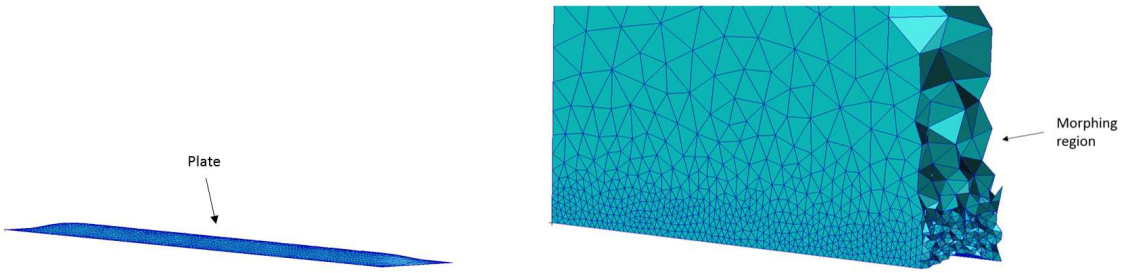


Figure 36: Morphing region – Plate and FEM_{CFD}

The structural wetted surface is made by 3135 nodes (18810 DOF) and the aerodynamic wetted surface by 1956 nodes (11736 DOF). The Spline module interpolates at each time step the aerodynamic pressure load from the CFD to the structural model, Fig. 37, and interpolates back the displacement from the structural to the aerodynamic solver, Fig. 38.

Figure 37: Load interpolation – aero mesh (left) and structural mesh (right)

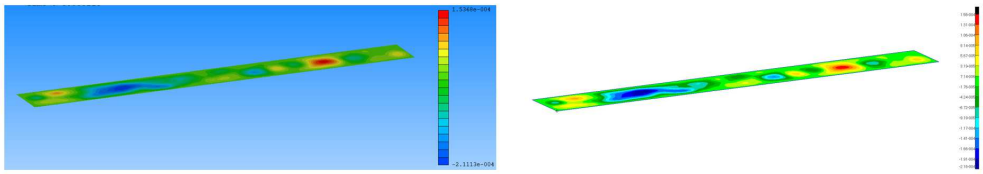


Figure 38: Displacement interpolation – aero mesh (left) and structural mesh (right)

In order to quantify the effect of the SPLINE module with respect to the overall computational time, two different simulations have been carried out and the computational time compared. One pure structural simulation has been run without the coupling to the CFD solver and one other with the coupling. For both cases the excitation given by the turbulent boundary layer has been included and the analysis performed with a time step of 1.E-5.
The time required by the CFD solver for each iteration has been removed from the computation in order to focus only on the time needed by the SPLINE module to interpolate load and displacement between the two interfaces and calculate the morphing of the CFD domain.
It can be observed that the fluid-structure simulation (blue dots) takes slightly longer to perform each iteration, especially at the beginning of the job, than the structural simulation without the coupling to the CFD solver (red dots), Fig 39.

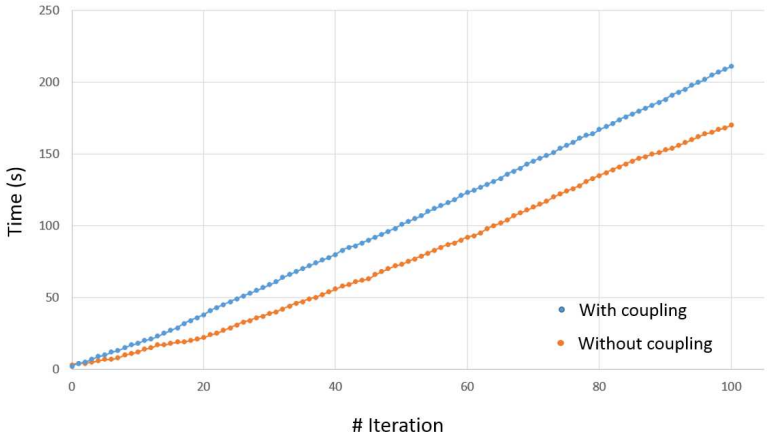


Figure 39: SOL 400 – CPU time (no CFD time considered)

The small difference between the two curves, which seems to stabilize after the first 40 iterations, is caused by the number of cycles carried out within a time step by the structural solver that increases in the case of the fluid-structure interaction simulation. In the case of FSI Nastran needs more cycles than the case of a pure structural simulation to converge Fig. 40. This results is probably caused by the effect of the combination of the mechanical (turbulent boundary layer) and aerodynamic loading that let the structure deform more than the case without the aerodynamic load.

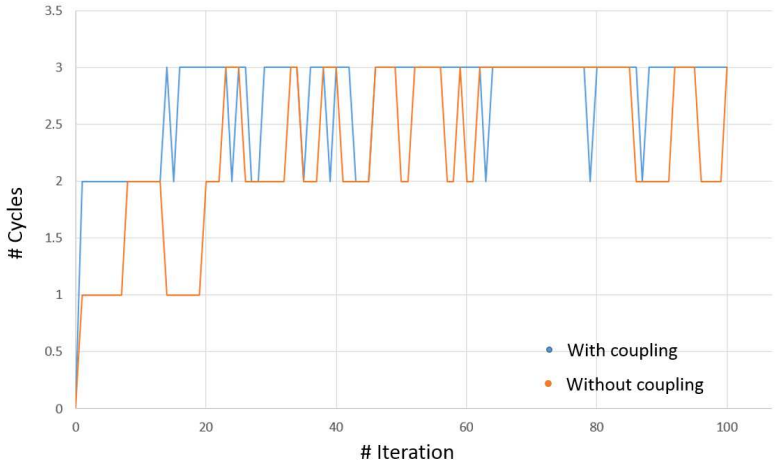


Figure 40: N. of sub-iterations within each time step

The f04 result file of the FSI simulation helps better understand the time got by the SPLINE module to perform one load interpolation during the simulation, Fig 41.

14:18:49	0:42	6.0	0.0	13.0	0.1	SPLINE	222	FORTIO BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	116	UDMSRV BEGN	Pack
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	118	FORTIO BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	124	INPUTT4 BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	125	MPYAD BEGN	$\{F_G\} = [G_{KG}^P]^T\{F_K\}$
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	127	FORTIO BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	129	FORTIO BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	138	PARAML BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	139	MATGEN BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	140	PARTN BEGN	
14:18:49	0:42	6.0	0.0	13.0	0.0	SPLINE	141	UDMSRV BEGN	Unpack

Figure 41: SPLINE module extract – Load interpolation

The load interpolation is executed in less than 0.1 seconds to get the CFD load from SCTetra (LINE 116), compute the interpolation (LINE 125) and send the structural load to SOL 400 OpenFSI (LINE 141). About 2 seconds are required by the structural solver to solve the time step (slightly longer than in the case without the coupling), and give back the computed structural displacement to the SPLINE module (from 0:42 to 0:44). Less than 0.1 seconds are used by the SPLINE module to get the structural displacement (LINE 194) solve the displacement interpolation (LINE 219) and morphing of the CFD domain (LINE 220), and send data to the CFD solver, Fig 42.

14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	187	FORTIO BEGN	
14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	194	UDMSRV BEGN	Pack
14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	195	FORTIO BEGN	
14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	207	MERGE BEGN	
14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	213	ADD BEGN	
14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	219	MPYAD BEGN	$\{u_K\} = [G_{KG}^D]\{u_G\}$
14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	220	FBS BEGN	$[K_{LL}]\{X\} = \{P_f\}$
14:18:51	0:44	6.0	0.0	13.1	0.0	SPLINE	221	UDMSRV BEGN	Unpack

Figure 42: SPLINE module extract – Load interpolation

Results from a fluid-structure simulation of a 0.01 second with a fixed time step of 1.E-5 are here presented.
The structure presents high deformation values especially at the interface with the rigid plate where it is fixed. The plastic strain is high in the area close to the attachment with the rigid plate, where the flexible structure is supported to, Fig 43.

Figure 43: Plastic strain

The maximum displacement and acceleration (envelope) have the highest values close to the "leading edge", Fig 44.

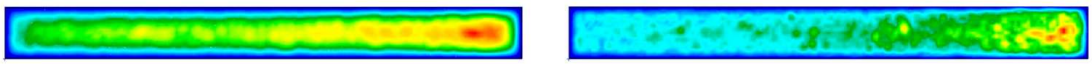


Figure 44: Magnitude of displacement (left) and acceleration (right) – Maximum

Typical waves that characterize the nonlinear panel flutter can be well observed, Fig 45 and Fig 46.

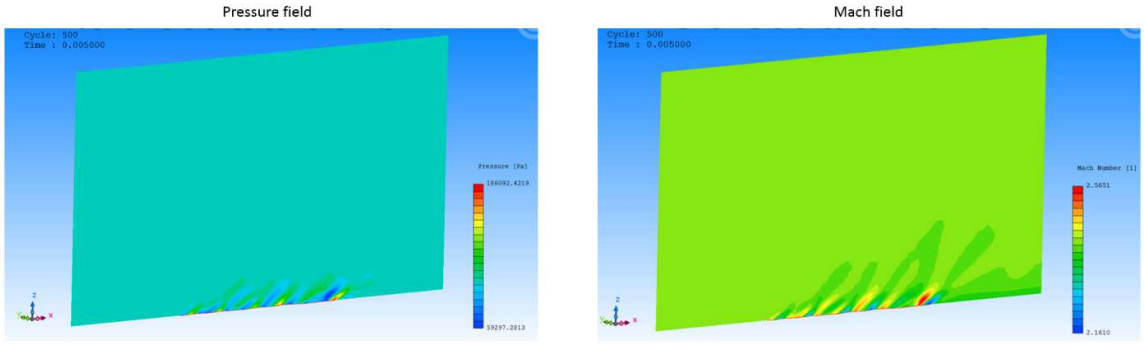


Figure 45: Pressure and Mach distribution

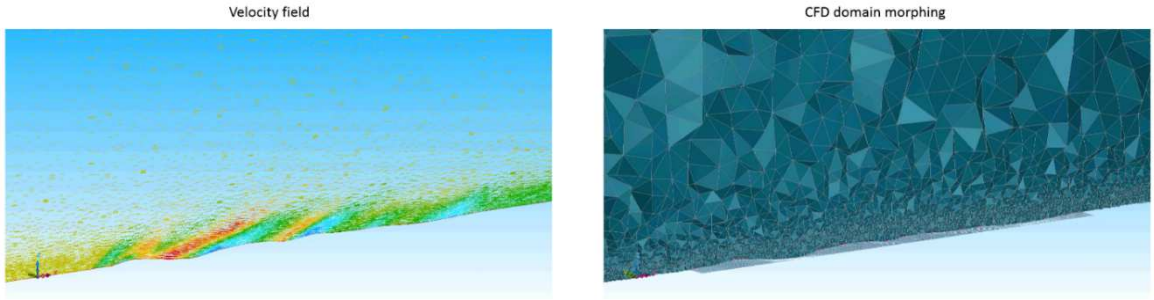


Figure 46: Velocity field and mesh morphing

Even though no evident instability with large displacement was observed but rather very large plastic deformations the proposed approach allowed to predict the LCO phenomena, Fig 47.

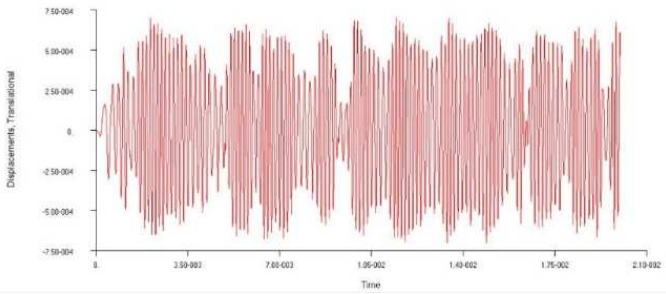


Figure 47: z-Displacement of a node in the symmetry plane (LCO)

4 CONCLUDING REMARKS

A novel open aero-structure interface has been presented to allow for accurate nonlinear FSI simulations for highly flexible structures. The new main features developed in the methodology are: a general coupling interface to enable the communication between the Nastran solver and any mesh-based CFD solver; a new SubDMAP SPLINE solution with a proper IDL that allows to access both structural and aerodynamic data thanks to the aid of a specific UDMSRV module developed for that purpose; The SPLINE module runs as an independent job and performs the interpolation at the aero and structural interface and the morphing of the CFD domain; Two different morphing algorithms have been implemented within the SPLINE module with novel functionalities to provide additional control on the morphing and improve the robustness and efficiency. One is based on the FEM analogy while another one is a hybrid approach that combines the FEM analogy with the RIS technique.
The methodology has been validated on the study of the nonlinear panel flutter of a supersonic plate.
Future work will address the proposed approach to the prediction of the nonlinear steady-state response of the uCRM [14].

5 REFERENCES

[1] Anon., MSC Nastran 2016 Non Linear Users Guide SOL400, MSC.Software Corporation, 2 MacArthur Place, Santa Ana, CA, USA, 2009.

[2] Di Vincenzo, F. G., Linari, M., Mohdzawawi, F., and Morlier, J. “Nonlinear Aeroelastic Steady Simulation Applied to Highly Flexible Blades for MAV”, 17th International Forum on Aeroelasticity and Structural Dynamics (IFASD), Como, Italy, 25–28 June 2017.

[3] MSC Nastran 2018 DMAP Programmer’s Guide , MSC.Software Corporation, 2 MacArthur Place, Santa Ana, CA, USA, 2009.

[4] Robinson M., Programming DMAP in MSC.Nastran™, 2012.

[5] https://www.cradle-cfd.com/

[6] Farhat, C., & Lesoinne, M. (2000). Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. Computer methods in applied mechanics and engineering, 182(3), 499-515.

[7] MSC.Software SOSP-73.13, “MSC Simulation Component Architecture Build System Guide”.

[8] MSC.Software Corp., 2 MacArthur Place, Santa Ana, CA 92707, USA, “Aeroelastic Analysis User’s Guide”, 2016.

[9] Di Vincenzo F. G., Castrichini A. Hybrid Static Aeroelasticity Toolkit (HSA). MSC Software, Toulouse, France (2013).

[10] Riso, C., Di Vincenzo, F. G., Ritter, M., Cesnik, C. E. S., and Mastroddi. “Nonlinear Aeroelastic Trim of Very Flexible Aircraft Described by Detailed Models”, Journal of aircraft, Vol 0, 2018.

[11] Riso, C., Di Vincenzo, F. G., Ritter, M., Cesnik, C. E. S., and Mastroddi, F. “A FEM-Based Approach for Nonlinear Aeroelastic Trim of Highly Flexible Aircraft,” 17th International Forum on Aeroelasticity and Structural Dynamics (IFASD), Como, Italy, 25–28 June 2017.

[12] Bosco, E. Lucchetti, A., Trapier, S., Di Vincenzo, F. G., Gourdain, N., Morlier, J.. “Wind-tunnel testing for validation of a method for nonlinear Fluid/Structure interaction using surrogate models”, 17th International Forum on Aeroelasticity and Structural Dynamics (IFASD), Como, Italy, 25–28 June 2017.

[13] Bosco, E. Lucchetti, A., Trapier, S., Di Vincenzo, F. G., Gourdain, N., Morlier, J.. ”Transient Fluid/Structure interaction approach using surrogate models: Industrial application to aircraft fairing vibration excited by engine efflux.” Scitech AIAA, 4–8 January 2016, San Diego, California.

[14] https://commonresearchmodel.larc.nasa.gov/

COPYRIGHT STATEMENT