Discrete Data Transfer Technique for Fluid–Structure Interaction

Jamshid A. Samareh*
NASA Langley Research Center, Hampton, VA 23681

This paper presents a general three-dimensional algorithm for data transfer between dissimilar meshes. The algorithm is suitable for applications of fluid-structure interaction and other high-fidelity multidisciplinary analysis and optimization. Because the algorithm is independent of the mesh topology, we can treat structured and unstructured meshes in the same manner. The algorithm is fast and accurate for transfer of scalar or vector fields between dissimilar surface meshes. The algorithm is also applicable for the integration of a scalar field (e.g., coefficients of pressure) on one mesh and injection of the resulting vectors (e.g., force vectors) onto another mesh. The author has implemented the algorithm in a C++ computer code. This paper contains a complete formulation of the algorithm with a few selected results.

Nomenclature

A	=	area
B	=	finite element basis functions
d	=	distance
f	=	load vectors
F	=	data matrix
K	=	stiffness matrix
L	=	length of a 2-sided element
M	=	moments
n	=	surface unit normal vector
p	=	projection of CFD mesh points onto CSM mesh
P	=	pressure loads
r	=	element coordinates
R	=	mesh coordinates
T	=	transformation matrix
U	=	virtual work
V	=	design variables
x, y, z	=	coordinates
δ	=	deflection vectors
ξ_1, ξ_2	=	element parametric coordinates

Subscripts

e	=	element
F	=	fluid
i, j, k	=	source point number, element number, and target point number, respectively
S	=	structures
T	=	total
$1, 2$	=	mesh numbers

Superscripts

S, T	=	source and target meshes

* Senior Research Engineer, Vehicle Analysis Branch, MS 451, Associate Fellow.

I. Introduction

Accurate data transfer among various disciplines is a key ingredient in any high-fidelity multidisciplinary analysis and optimization. For example, the strong coupling between the external flow and the aircraft structure can prompt physically important phenomena. Correct modeling of these complex aeroelastic phenomena requires an accurate coupling of computational structural mechanics (CSM) and computational fluid dynamics (CFD) for a flexible structure. In a multidisciplinary environment, various disciplines share the same geometry, but the models (e.g., CSM and CFD) have dissimilar meshes. In addition, the data from one discipline must be available to all other disciplines. The data may be scalar (e.g., pressure and temperature), vector (e.g., deflection and heat transfer), or integrated quantities (e.g., aerodynamic and thermal loads). The data transfer process may be subjected to additional constraints, such as conservation of forces, moments, and energy.

Samareh and Bhatia[1] provide a review of existing data transfer algorithms, which range from earlier algorithms like FLEXSTAB[2] at Boeing in early seventies to Dassault Aviation work[3-4] in the eighties. In recent years, various researchers have examined the data transfer process primarily for aeroelastic analysis. Discrepancies and dissimilarities in the geometry and the mesh models are two potential sources of error in the data transfer process. One source of error may arise if the models have dissimilar levels of geometry details. For example, a CFD mesh generally resembles the true outer model line geometry of the aircraft; the CFD mesh includes details such as pylons, nacelles, flaps, and slats. However, a CSM mesh generally represents only the major structural components, such as the wing box. Flaps and slats are represented either by a few simple beam elements or are completely excluded. Tzong et al.[5] and Kapania and Bhardwaj[6] present data transfer algorithms that are based on the finite element (FE) method, in which the virtual work is used to transfer the aerodynamic pressures onto a CSM mesh. Then, the displacements are transferred back to the CFD mesh through the reciprocal theorem. Kapania and Bhardwaj[6] were successful in using a simplified version of this algorithm for several wings.

Brown[7] adds virtual elements in the CSM model to cover the discrepancies in the geometry definition between CSM and CFD models. These virtual elements add neither stiffness nor mass. As pointed out by Cebral and Löhner[8], the generation of virtual elements is an unnecessary complication, particularly for complex geometries.

Smith, Hodges, and Cesnik[9] provide excellent overviews of six data transfer methods between CFD and CSM disciplines. These methods are infinite-plate spline (IPS), multiquadric biharmonic (MQ), nonuniform B-spline (NUBS), thin-plate spline (TPS), finite-plate spline (FPS), and inverse isoparametric mapping (IIM). The IPS method is the basis for the popular surface splines method, which is available in most commercial aeroelastic analysis tools (e.g., MSC NASTRAN[†]). The IPS method interpolates a function of two independent variables and requires noncoincident mesh points. Smith, Hodges, and Cesnik[9] recommend further study of IIM and NUBS. They indicate that IIM shows great promise for two-dimensional applications and needs extension to three dimensions.

Clutter[10] and Send[11] extend NUBS to three dimensions. One major limitation with most NUBS implementation is that the input data must be a structured (regular) mesh. This limitation forces the data, at best, to be approximated in most realistic cases. Samareh[12] presents a method that removes this limitation, and the method uses a non-uniform rational B-spline (NURBS) representation for data transfer among various disciplines. This method is a general three-dimensional, least-squares representation, which removes the requirement for the structured input mesh and can handle multiple coincident points. Another advantage of this method is that the users have control over the tradeoff between smoothness and accuracy.

Murti and Valliappan[13] present an IIM algorithm for a two-dimensional model, and Pidaparti[14] further refines the IIM algorithm. The refined algorithm uses the FE shape functions to interpolate the coordinates, pressure, and displacement vectors. Because the FE shape functions satisfy a positivity constraint, the process will maintain local extrema. Maman and Farhat[15] and Farhat, Lesoinne, and LeTallec[16] outline a consistent interpolation algorithm similar to IIM for transferring information between two dissimilar meshes. The local interpolation is computed by projecting one mesh onto another. Cebral and Löhner[17] present a variation of IIM that guarantees conservation of forces. They use a Galerkin method to solve for the pressure on the CSM mesh. They also use an adaptive Gaussian integration technique to improve the accuracy. Farhat, Lesoinne, and LeTallec[16] also present a variation of their original algorithm that guarantees conservation of forces. We used a variation of this algorithm for our current study.

The accuracy of the data transfer process for integrated quantities (e.g., forces, moments, and energy) depends on the consistency of data transfer as well as other constraints, such as conservation of forces and moments. For example, the following equation defines consistent load vectors for the structural analysis on a CSM mesh as:

[†] MSC NASTRAN is a registered trademark of the McNeal-Schwendler Corporation

$$\{\bar{f}_e\} = \int_A \{B\}^T \{P\} \bar{n} dA \tag{1}$$

where $\bar{f}_e$ is the elemental load vector, B is the FE shape function, P is the nodal pressure, $\bar{n}$ is the unit surface normal on the CSM mesh, and dA is the infinitesimal surface element on the CSM mesh. Because the above equation uses the same shape functions as used to calculate the element stiffness matrix, the equation guarantees a consistent loading.[18] There are several possible problems with using the above equation. First, the aerodynamic load may have a large variation within a single CSM element, such that the shape functions are inadequate to capture the load variation across the element. Second, the FE shape functions may be unavailable for some commercial CSM codes. Third, the local normal vectors for a CSM mesh are generally different to a CFD mesh, primarily due to differences in discretization. In its present form, Eq. (1) does not guarantee conservation of forces and moments. An alternative approach to Eq. (1) is to integrate the loads on the CFD mesh and inject the resulting force vectors to the CSM mesh. This approach guarantees conservation of forces and moments. One limitation of this method is that the CFD mesh must have equal or higher resolution than the CSM mesh. Refining the CFD surface mesh (e.g., splitting quadrilateral into four elements) and interpolating the corresponding data will alleviate this problem without refining the entire field mesh. The next section presents a general algorithm, followed by some selected results and summary.

II. Data Transfer Algorithm

We can represent the data transfer between two dissimilar meshes in a matrix form as:

$$\{F_2\} = [T_{21}]\{F_1\} \tag{2}$$

where matrix $\{F_1\}$ represents the input data on the source mesh, matrix $\{F_2\}$ represents the output data on the target mesh, and matrix $[T_{21}]$ is a transformation matrix. For example, $\{F_1\}$ could represent the aerodynamic loads defined on a CFD mesh and transferred to a CSM mesh as $\{F_2\}$. Generally, the transformation matrices are large and sparse. If the transformation matrix $[T_{21}]$ is independent of the shape changes, then we can calculate $[T_{21}]$ once and use it as long as there is no change in the mesh connectivity.

The use of a transformation matrix simplifies the integrated analyses such as aeroelastic calculation. The aeroelastic calculation has four distinct steps: a) calculate aerodynamic loads on the CFD mesh (F_F), b) transfer aerodynamic loads to the CSM mesh (F_S), c) calculate the aeroelastic deflections on the CSM mesh (δ_S), and d) transfer the aeroelastic deflection to the CFD mesh (δ_F) to recalculate and update the aerodynamic loads. Equation (3) summaries this process as:

$$\begin{aligned} \{\bar{F}_F\} &= \{\bar{F}_F(\bar{R}_F + \bar{\delta}_F)\}, && \text{step (a)} \\ \{\bar{F}_S\} &= [T_{SF}]\{\bar{F}_F\}, && \text{(b)} \\ [K]\{\bar{\delta}_S\} &= \{\bar{F}_S\}, && \text{(c)} \\ \{\bar{\delta}_F\} &= [T_{FS}]\{\bar{\delta}_S\}, && \text{(d)} \end{aligned} \tag{3}$$

where $\bar{F}_F$ is the aerodynamic load vector on the CFD mesh, $\bar{R}_F$ are coordinates of the CFD mesh, $\bar{\delta}_F$ are aeroelastic deflections on the CFD mesh, $\bar{F}_S$ is the aerodynamic load vector on the CSM mesh, $\bar{\delta}_S$ are the aeroelastic deflections on the CSM mesh, and $[T]$ is the transformation matrix. Combining Eqs. 3a with 3d and 3b with 3c results in,

$$\begin{aligned} \{\bar{F}_F\} &= \{\bar{F}_F(\bar{R}_F + [T_{FS}]\{\bar{\delta}_S\})\} \\ [K]\{\bar{\delta}_S\} &= [T_{SF}]\{\bar{F}_F\} \end{aligned} \tag{4}$$

For a linear structure without rigid body degrees-of-freedom, we can further simplify Eq. (4) to

$$\{\bar{F}_F\} = \{\bar{F}_F(\bar{R}_F + [T_{FS}][K]^{-1}[T_{SF}]\{\bar{F}_F\})\} \tag{5}$$

We can extend this formulation to applications with rigid body degrees-of-freedom. Through Eq. 5, CFD codes can be decoupled from a linear CSM code.

If the transformation matrix is independent of the shape changes, then the formulation is especially appealing for sensitivity analysis used in a gradient-based optimization. For example, we can use the following equation to transfer the CFD load sensitivity to a CSM mesh as,

$$\frac{\partial\{\bar{F}_S\}}{\partial V} = [T_{SF}]\frac{\partial\{\bar{F}_F\}}{\partial V}$$

where V is the design variable vector, and $[T_{SF}]$ is a constant transformation matrix.

We can reformulate most data transfer algorithms and present them in a matrix form. Next section provides a reformulation of the algorithm presented by Farhat, Lesoinne, and LeTallec[16].

III. Discrete Data Transfer

The algorithm presented here is a derivative of the IIM algorithm and the algorithm presented in Farhat, Lesoinne, and LeTallec[16]. The discrete data transfer process consists of three steps: 1) find the nearest source element for every target point, 2) calculate the mapping coefficients for every target point and its corresponding (nearest) source element, and 3) interpolate/inject from the source element to the corresponding target point. We can use the results from steps 1-2 for multiple data transfer applications so long the topologies of the source and the target meshes are unchanged.

The first step requires a spatial proximity search, which is the most time-consuming part of the entire process. We must first find a nearest source element to a target point (see Fig. 1). One way to accomplish this step is to project every target point to every source element. This is an exhaustive search, which has the complexity of $O(N^2)$ and is prohibitively expensive. Samet[19] presents several methods and data structures suitable for the proximity search. We use the alternative digital tree (ADT) method introduced by Bonet and Peraire[20], which is similar to an octree method. The ADT method reduces the search complexity to $N\,O(\log(N))$, which is far more efficient than the exhaustive search. After application of ADT, we will have a handful of candidate source elements for every target point.

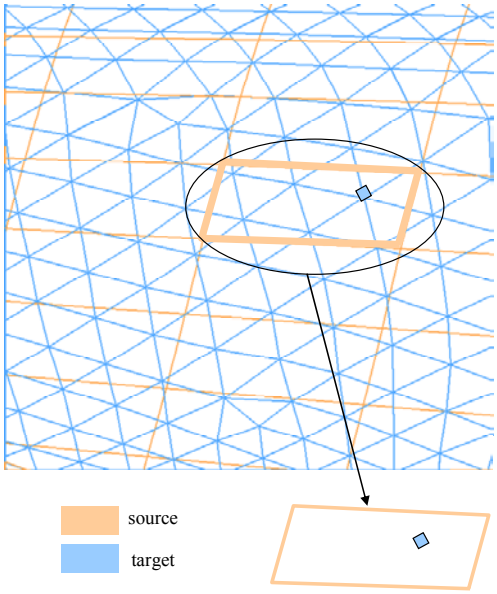
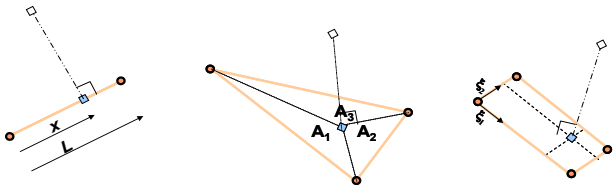


Figure 1. Source and Target Meshes.

For every target point and its candidate source elements, we need to find a candidate point on the source element that is the nearest to the target point. Then, we select the closest candidate point and its corresponding element. In order to accomplish this, we need to have a complete definition for the source element shape. We use finite element basis functions for this purpose. Using these functions, one can define element shape as:

$$\bar{R}_j^s(\xi_1,\xi_2) = \sum_i \bar{r}_{i,j}^s B_i(\xi_1,\xi_2), \quad \text{where } \sum_i B_i(\xi_1,\xi_2) = 1 \tag{6}$$

The term $\bar{R}_j^s$ represents source element shape, $\bar{r}_{i,j}^s$ are source element nodes, ξ_1 and ξ_2 are the element parametric coordinates, and B_i are the element basis functions. Standard books on finite element analysis contain detailed discussions on finite element basis functions and their properties. Figure 2 shows the basis functions for bar, triangle, and quadrilateral elements.



2-Sided Elements

$$B_1(\xi_1,\xi_2) = \xi_1 = \frac{x}{L}$$
$$B_2(\xi_1,\xi_2) = \xi_2 = 1-\xi_1$$

3-Sided Elements

$$B_1(\xi_1,\xi_2) = \frac{A_1}{A_T} = \xi_1$$
$$B_2(\xi_1,\xi_2) = \frac{A_2}{A_T} = \xi_2$$
$$B_3(\xi_1,\xi_2) = \frac{A_3}{A_T} = 1-(\xi_1+\xi_2)$$
$$A_T = A_1 + A_2 + A_3$$

4-Sided Elements

$$B_1(\xi_1,\xi_2) = (1-\xi_1)(1-\xi_2)$$
$$B_2(\xi_1,\xi_2) = \xi_1(1-\xi_2)$$
$$B_3(\xi_1,\xi_2) = \xi_1\xi_2$$
$$B_4(\xi_1,\xi_2) = (1-\xi_1)\xi_2$$

Figure 2: Elements and Basis Functions.

As shown in Fig. 2, the nearest point on a source element to a target point has the minimum distance between the two points. Equation 7 defines the distance from a target point ($\bar{R}_k$) to any point on the source element (j) as a function of the source element parameters:

$$d^2(\xi_1^k,\xi_2^k) = \left\| \bar{R}_k - \sum_i \bar{r}_{i,j}^s B_i(\xi_1^k,\xi_2^k) \right\| \tag{7}$$

The nearest location on the source element to the target point can be determined by finding the minimum distance:

$$\underset{\xi_1^k,\xi_2^k}{Min}\left(d^2(\xi_1^k,\xi_2^k)\right)$$
$$\frac{\partial(d^2)}{\partial\xi_1^k} = 0 \quad \Rightarrow \quad [\bar{R}_k - \sum_i \bar{r}_{i,j}^s B_i(\xi_1^k,\xi_2^k)] \bullet [\sum_i \bar{r}_{i,j}^s \frac{\partial B_i(\xi_1^k,\xi_2^k)}{\partial\xi_1^k}] = 0 \tag{8}$$
$$\frac{\partial(d^2)}{\partial\xi_2^k} = 0 \quad \Rightarrow \quad [\bar{R}_k - \sum_i \bar{r}_{i,j}^s B_i(\xi_1^k,\xi_2^k)] \bullet [\sum_i \bar{r}_{i,j}^s \frac{\partial B_i(\xi_1^k,\xi_2^k)}{\partial\xi_2^k}] = 0$$

For bar and triangular elements, Eq. (8) results in a system of linear equations. For quadrilateral elements, Eq. (8) results in a system of quasi-linear equations, and we use a damped Newton-Raphson technique to solve them. For example, Eq. (8) can be rewritten for a triangular element ($B_1 = \xi_1$, $B_2 = \xi_2$, and, $B_3 = 1 - \xi_1 - \xi_2$) as:

$$\begin{aligned}(\bar{r}_{1,j}^s - \bar{r}_{3,j}^s)\bullet(\bar{r}_{1,j}^s - \bar{r}_{3,j}^s)\xi_1^k + (\bar{r}_{2,j}^s - \bar{r}_{3,j}^s)\bullet(\bar{r}_{1,j}^s - \bar{r}_{3,j}^s)\xi_2^k &= (\bar{R}_k - \bar{r}_{3,j}^s)\bullet(\bar{r}_{1,j}^s - \bar{r}_{3,j}^s) \\ (\bar{r}_{1,j}^s - \bar{r}_{3,j}^s)\bullet(\bar{r}_{2,j}^s - \bar{r}_{3,j}^s)\xi_1^k + (\bar{r}_{2,j}^s - \bar{r}_{3,j}^s)\bullet(\bar{r}_{2,j}^s - \bar{r}_{3,j}^s)\xi_2^k &= (\bar{R}_k - \bar{r}_{3,j}^s)\bullet(\bar{r}_{2,j}^s - \bar{r}_{3,j}^s)\end{aligned} \tag{9}$$

where $r^s_{i,j}$ are the nodes of source element j. Solving this linear system results in ξ_1^k and ξ_2^k, which are the parametric coordinates of a point nearest to the target point k on the source element j. This concludes the second step in our process.

We have now sufficient information to transfer scalar or vector data using the source element definition as

$$f_k(\xi_1^k,\xi_2^k) = \sum_i f_i^s B_i(\xi_1^k,\xi_2^k) \tag{10}$$

where f_i^s is the data defined on the source element, and f_k is the interpolated data for point k on the target mesh. We can represent Eq. (10) in a matrix form as

$$[F_T] = [T_{TS}][F_S] \text{ where } \quad T_{ki} = B_i(\xi_1^k,\xi_2^k), \sum_i T_{ki} = 1 \tag{11}$$

For example, F_S could represent the temperature distribution on a CFD mesh, and F_T would be the resulting interpolated temperature distribution on a CSM mesh.

This algorithm could be easily adapted for an aeroelastic analysis, where the aeroelastic deflection is transferred from a CSM mesh to a CFD mesh and aerodynamic loads from a CFD mesh to a CSM mesh. To interpolate the aeroelastic deflection from a CSM mesh to a CFD mesh, we can rewrite Eq. (11) as:

$$[\bar{\delta}_{CFD}] = [T_{CFD-CSM}][\bar{\delta}_{CSM}] \tag{12}$$

where $\bar{\delta}_{CFD}$ and $\bar{\delta}_{CSM}$ are the aeroelastic deflections on the CFD and the CSM meshes, respectively. There are two approaches to transfer CFD loads to a CSM mesh: 1) interpolate pressure onto the CSM mesh and integrate the pressure on the CSM mesh, and 2) integrate pressure on the CFD mesh and then transfer the resulting load vectors onto the CSM mesh. The latter approach guarantees conservation of forces and moments, but not the former approach. We use virtual work to transfer the aerodynamic loads to CSM mesh. The virtual work, product of force and deflection, on the CSM mesh (U_{CSM}) must be equal to the virtual work on the CFD mesh (U_{CFD}) as

$$\begin{gathered} U_{\mathrm{CSM}} = \bar{F}_{CSM}^{T}\bar{\delta}_{\mathrm{CSM}},\ U_{CFD} = \bar{F}_{CFD}^{T}\bar{\delta}_{\mathrm{CFD}}, \\ U_{CSM} = U_{CFD} \\ \bar{F}_{CSM}^{T}\bar{\delta}_{\mathrm{CSM}} = \bar{F}_{CFD}^{T}\bar{\delta}_{\mathrm{CFD}} \end{gathered} \tag{13}$$

Substituting Eq. (12) into Eq. (13) and rearranging terms result in:

$$\begin{gathered} [\bar{F}_{CSM}^{T}][\bar{\delta}_{\mathrm{CSM}}] = [\bar{F}_{CFD}^{T}][T_{CFD-CSM}][\bar{\delta}_{CSM}] \\ [\bar{F}_{CSM}^{T}] = [\bar{F}_{CFD}^{T}][T_{CFD-CSM}] \\ [\bar{F}_{CSM}] = [T_{CFD-CSM}]^{T}[\bar{F}_{CFD}] \end{gathered} \tag{14}$$

Equation (14) satisfies the reciprocity relation, which allows us to use the same transformation matrix to transfer deflections and loads as

$$\begin{gathered} \left[\bar{\delta}_{CFD}\right] = [T_{CFD-CSM}][\bar{\delta}_{CSM}] \\ \left[\bar{F}_{CSM}\right] = [T_{CFD-CSM}]^{T}[\bar{F}_{CFD}] \end{gathered} \tag{15}$$

In addition to the virtual work, Eq. (15) satisfies conservation of forces as:

$$\begin{aligned} \sum_k \bar{F}_k^{CFD} &= \sum_i \bar{F}_i^{CSM} \\ &= \sum_i \left[T^{\mathrm{CFD-CSM}}\right]^{T}\left[\bar{F}^{CFD}\right] \\ &= \sum_i \sum_k T_{\mathrm{ik}}^{\mathrm{CFD-CSM}} \bar{F}_k^{CFD} \\ &= \sum_k \bar{F}_k^{CFD} \sum_i T_{\mathrm{ki}}^{\mathrm{CFD-CSM}}, \quad \sum_i T_{\mathrm{ki}}^{\mathrm{CFD-CSM}} = 1 \\ &= \sum_k \bar{F}_k^{CFD} \end{aligned} \tag{16}$$

Therefore, we have proven that Eq. (15) maintains virtual work and conservation of forces. Next, we will look at conservation of moments. Although the source and target meshes are created from the same geometry model, there still exists a gap between the two meshes due to different levels of mesh discretization. This gap creates a moment deficit (ΔM^{CSM}), which needs to be taken into account. The conservation of moments requires that the moments on CSM and CFD meshes to be equal as:

$$\sum_i \bar{r}_i^{CFD} \times \bar{F}_i^{CFD} = \sum_k \left\{ \bar{r}_k^{CSM} \times \bar{F}_k^{CSM} + \Delta M_k^{CSM} \right\} \tag{17}$$

We can expand this equation as:

$$\begin{aligned} \sum_i \bar{r}_i^{CFD} \times \bar{F}_i^{CFD} &= \sum_k \bar{r}_k^{CSM} \times \left(\left[T^{\text{CFD-CSM}} \right]^T \left[\bar{F}^{CFD} \right] \right) + \sum_k \Delta M_k^{CSM} = \sum_k \bar{r}_k^{CSM} \left[T^{\text{CFD-CSM}} \right]^T \times \left[\bar{F}^{CFD} \right] + \sum_k \Delta M_k^{CSM} \\ &= \sum_k \left\{ \sum_i \left\{ T_{ik}^{\text{CFD-CSM}} \bar{r}_k^{CSM} \times \bar{F}_i^{CFD} \right\} \right\} + \sum_k \Delta M_k^{CSM} \\ &= \sum_i \left\{ \sum_k T_{ik}^{\text{CFD-CSM}} \bar{r}_k^{CSM} \right\} \times \bar{F}_i^{CFD} + \sum_k \Delta M_k^{CSM} = \sum_i \bar{p}_i^{CFD} \times \bar{F}_i^{CFD} + \sum_k \Delta M_k^{CSM} \end{aligned} \tag{18}$$

where $\bar{p}_i^{CFD}$ is the projection of CFD mesh points ($\bar{r}_i^{CFD}$) onto the CSM mesh. As a result,

$$\begin{aligned} \sum_k \Delta M_k^{CSM} &= \sum_i (\bar{r}_i^{CFD} - \bar{p}_i^{CFD}) \times \bar{F}_i^{CFD} \\ &= \sum_i \Delta M_i^{CFD} \end{aligned} \tag{19}$$

The term $(\bar{r}_i^{CFD} - \bar{p}_i^{CFD})$ represents the gap between the two meshes. We use the load transformation matrix to transfer nodal CFD moment deficits (ΔM_i^{CFD}) to CSM mesh (ΔM_k^{CSM}) as:

$$\left[\Delta M^{CSM} \right] = \left[T^{\text{CFD-CSM}} \right]^T \left[\Delta M^{CFD} \right] \tag{20}$$

This process maintains conservation of moments. In the next section, we present results for several cases.

IV. Results

The author has implemented the algorithm described in the previous section in a C++ computer program, and this program is available for distribution within the United States. We have applied the algorithm to a set of test cases, and Table 1 shows a summary of these results. We ran all test cases on a 3 GHz PC running Linux operating system. The conservation of forces and moments were satisfied within machine accuracy (10^{-12}) for all load transfer cases. It took less than ten seconds for a typical test case.

Table 1. Algorithm Performance

	Source		Target		
Test Cases	# of Points	# of Elements	# of Points	# of Elements	CPU Time (s)
1) MicroAirVehicle(Aero to FEM)	30656	60832	929	872	6.356
2) X43 (FEM to Aero)	986	1799	1776	3548	0.576
3a) X33 (Aero to Aero)	22987	22680	69051	137357	62.33
3b) X43 (Aero to FEM)	1776	3548	986	1799	0.48
4a) Ballute (Aero to FEM)	7151	8278	9904	17352	4.18
4b) Ballute (Aero to FEM)	9904	17352	7151	8278	3.42
4c) Ballute (Aero to FEM)	37144	69408	7151	8278	8.396
4d) Ballute (Aero to FEM)	143680	277632	7151	8278	29.12
4e) Ballute (Aero to FEM)	564948	1110528	7151	8278	113.5
5) Morphing Wing (Aero to FEM)	5700	5280	65	62	0.908
6) Mode Shape (FEM to Aero)	443	404	6201	6032	0.86
7) BWB (FEM 2 Aero)	38532	574324	18404	17472	17.621

The algorithm performance depends on the mesh sizes (number of points and elements for the source and target meshes), element definitions, discrepancies in mesh resolutions, gaps between two meshes, and a number of other factors. However, the performance approximately varies with the average of number of points for the source (*nps*) and target (*npt*) meshes. Figure 3 shows the algorithm performance (CPU time) versus average number of points. The dots represent the actual performance, and the line is the least-squares approximation of CPU time. One of the data points, shown as solid dot in Fig. 3, requires a larger CPU time due to the element definition and the shape. The results indicate that the performance generally scales linearly with the average number of points.

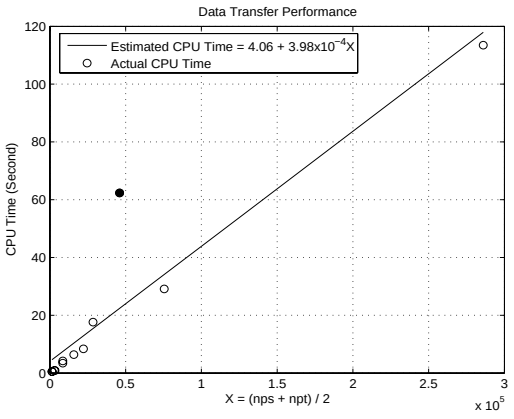


Figure 3. Performance Summary

The algorithm accuracy depends on the mesh resolution and the gaps between source and target meshes. The algorithm guarantees the load transfer accuracy, and our test results validated this fact. The algorithm maintains conservation of forces and moments within the machine accuracy ($\sim 10^{-12}$). We used a ballute (a combination of balloon and parachute used to decelerate a planetary vehicle) model to demonstrate the interpolation accuracy. We started with a coarse mesh on a ballute (Fig. 4a) with the sinusoidal function (Fig. 4b) superimposed on the entire surface as:

$$f_1(x,y,z) = \sin\left(\frac{x-800}{800}\right)\sin\left(\frac{y}{400}\right)\cos\left(\frac{z}{400}\right) \tag{21}$$

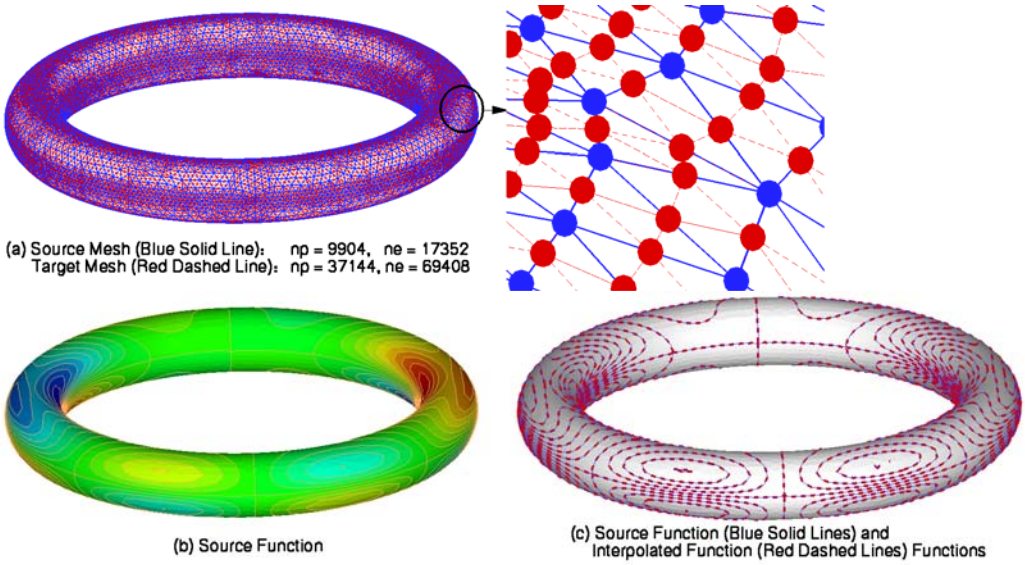


Figure 4: Interpolation Accuracy

We refined the coarse mesh by splitting each element to create our source mesh and function. We then used the coarse mesh (target) to interpolate function (f_2) from the fine (source) mesh. Figure 4c shows contours for source and resulting target meshes. If the process were error free, then (f_1) and (f_2) should be identical. The average difference between (f_1) and (f_2) was 6.5×10^{-17}, which is well below our machine accuracy.

The second test case is the aerodynamic load transfer from a CFD mesh to a FE mesh for a Micro Air Vehicle. Figure 5 shows: (a) CFD source mesh, (b) contours of coefficients of pressure on the source mesh, (c) aerodynamic load vectors on the CFD mesh, (d) target CSM mesh, and (e) the transferred load vectors on the target CSM mesh. The force vector distributions on the CSM and the CFD meshes are different due to different levels of discretization; however, the transfer process conserved forces, moments, and virtual work within the limits of the computer accuracy. It took 6.36 seconds of CPU time to run this case.

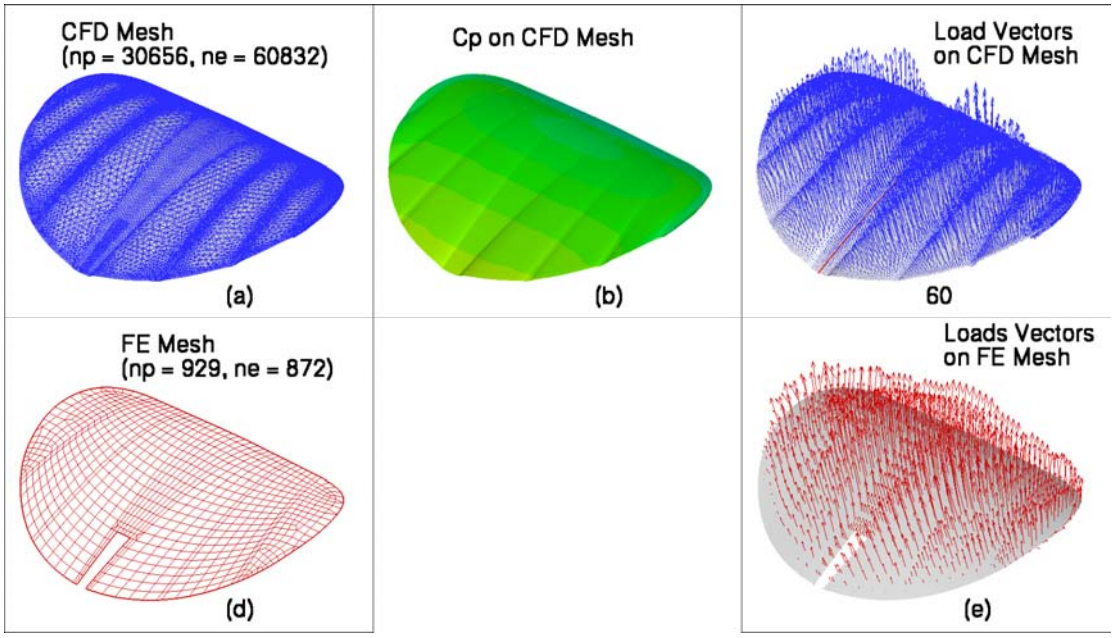


Figure 5. Load Transfer for a Micro Vehicle

The third test case is the interpolation of axial moment from a structured mesh to an unstructured mesh for an X33 model. Figure 6 shows: (a) the source structured mesh, (b) axial moment contours for the source mesh, (c) target mesh, and (d) contours of source and interpolated target axial moment. Figure 6d shows that the original source contours (solid blue lines) and interpolated target contour lines (red dashed lines). Both contour lines lie right on top of each other. It took 62.33 seconds of CPU time to run this case.

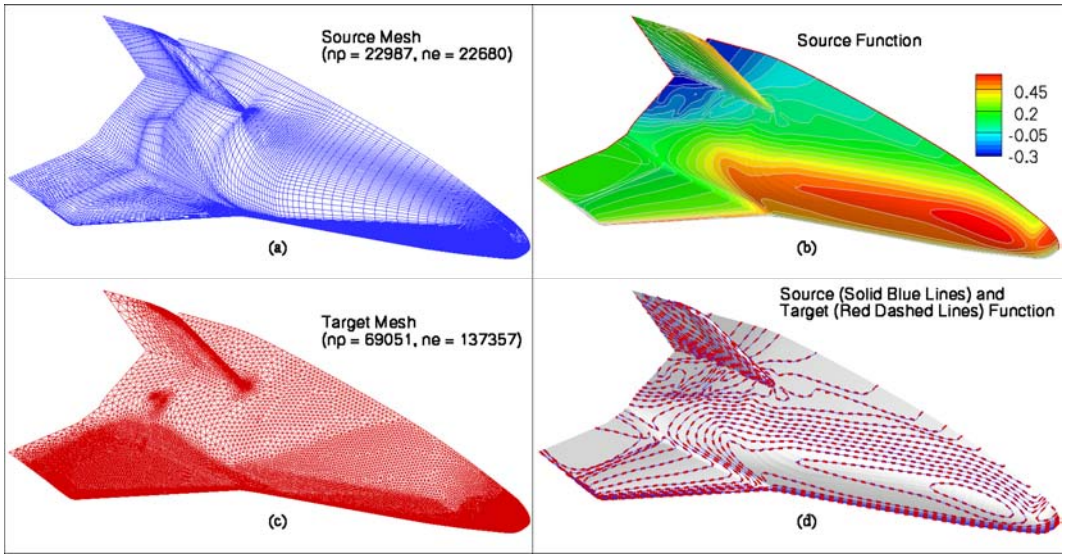


Figure 6. Pressure Interpolation on an X33.

The fourth test case is the load transfer for a morphing configuration. Figure 7 shows: (a) CFD mesh, (b) coefficients of pressure on the CFD mesh, (c) load vectors on the CFD mesh, (d) a simple mesh for multibody dynamics analysis (colors signify different zones), and (e) transferred load vectors on the target mesh. For clarity, the moment distribution is not shown in Fig. 7. As with the all other load transfer test cases, the process conserved forces, moments, and virtual work within the limits of the computer accuracy. It took 0.91 second of CPU time to run this case.

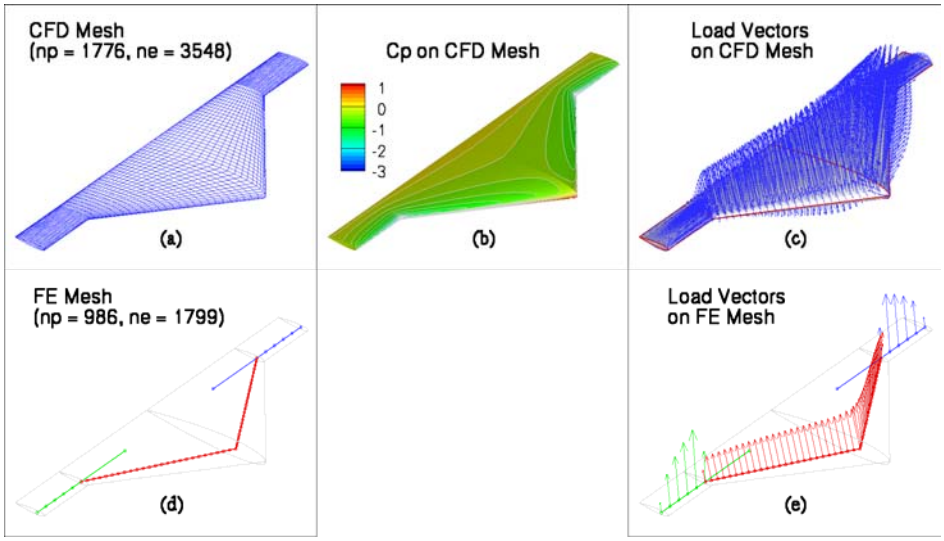


Figure 7. Load Transfer for a Morphing Vehicle.

The fifth test case is the deflection transfer for a blended wing body. Figure 8 shows: (a) undeflected and deflected FE meshes, (b) undeflected and deflected CFD meshes, (c) close-up view of deflected FE and CFD meshes. The transfer included all three components of the deflection vectors. The deflected CFD model is right on the deflected FE mesh. It took 17.62 seconds of CPU time to run this case.

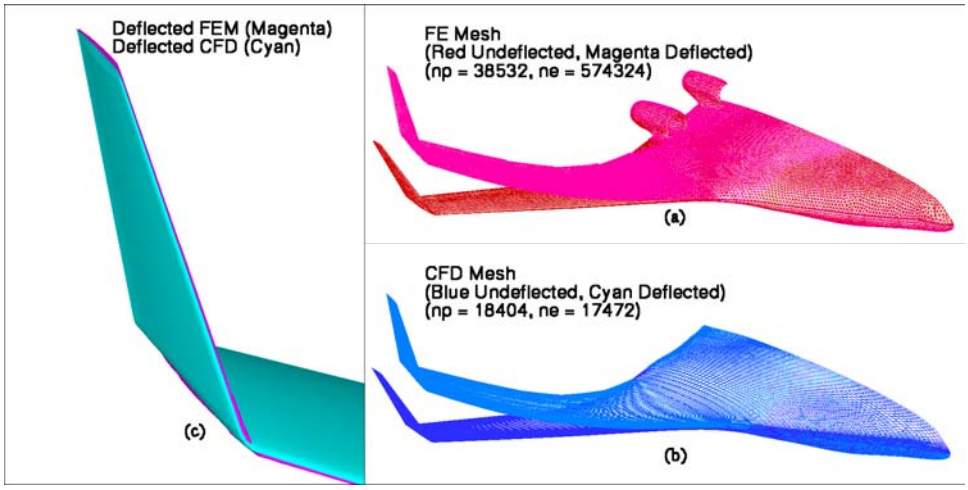


Figure 8. Deflection Transfer for a Blended Wing

The sixth and final test case is the aerodynamic load transfer for a ballute. Figure 9 shows: (a) CFD source mesh, (b) contours of coefficients of pressure on the source mesh, (c) target CSM mesh, and (d) the transferred load vectors on the target CSM mesh. The conservation of forces, moments, and virtual work were maintained within the limits of the computer accuracy. It took 4.18 seconds of CPU time to run this case.

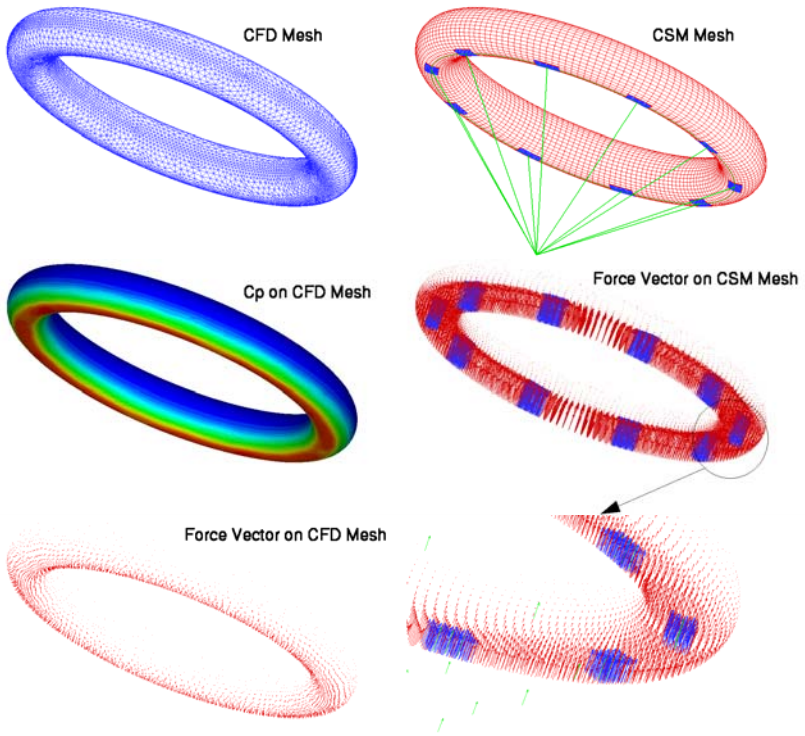


Figure 9. Load Transfer for a Ballute

V. Conclusions

We have presented an algorithm for data transfer between dissimilar meshes. The algorithm is suitable for high-fidelity multidisciplinary applications, where scalar or vector fields need to be transferred among various discipline. The paper includes a formulation for modeling fluid-structure interaction, and the formulation guarantees conservation of force and moments. The overall formulation is in a matrix form, which simplifies the algorithm integration with existing commercial software. The implementation is independent of the mesh topology, so we can treat structured and unstructured meshes in the same manner. The implementation also includes an advanced spatial search algorithm that substantially reduces the required computational resources.

We verified the algorithm accuracy through six test cases. The results for load transfer cases demonstrated that the algorithm maintained conservation of forces and moments. The test cases also verified the interpolation accuracy. The algorithm performance depends on the mesh sizes, element definitions, discrepancies in mesh resolutions, gaps between two meshes, and a number of other factors. However, the performance approximately varied linearly with the average of number of mesh points. It took less than ten seconds for a typical test case.

Acknowledgments

The author would like to thank Drs. Kumar Bhatia and Moeljo Hong of Boeing (Seattle) for a very productive collaboration during the past few years, and their hospitality and generosity during author's visits to Boeing. The author would also like to thank following NASA Langley Researchers: Paul Pao and Mercedes Reeves for providing

micro air vehicle data, Sasan Armand for providing the data for ballute, Karen Bibb for providing the X33 data, and Pawel Chwalowski for providing data for the morphing aircraft.

References

[1]Samareh, J. A., Bhatia, K. G., "A Unified Approach to Modeling Multidisciplinary Interactions," 8th AIAA/NASA/USAF/ISSMO Multidisciplinary Analysis and Optimization Conference, Multidisciplinary Analysis and Optimization, Long Beach, CA, 6-8 September 2000, AIAA Paper 2000-4704.

[2]Dusto, A. R., "A Method for Predicting the Stability Characteristics of an Elastic Airplane, FLEXSTAB Theoretical Description," NASA CR-114,712, Oct. 1974.

[3]Petiau, C. and Brun, S., "Trends in Aeroelastic Analysis of Combat Aircraft," AGARD, AD-P005855, Aug. 1987.

[4]Nicot, Ph., and Petiau, C., "Aeroelastic Analysis Using Finite Element Models," European Forum for Aeroelasticity and Structural Dynamics," Aachen, Germany, 1989.

[5]Tzong, G., Chen, H. H., Chang, K. C., Wu, T., and Cebeci, T., "A General Method for Calculating Aero-Structure Interaction on Aircraft Configurations," AIAA Paper 96-3982, Sept. 1996.

[6]Kapania, R. K., and Bhardwaj, M., "Aeroelastic Analysis of Modern Complex Wings," AIAA Paper 96-4011, Sept. 1996.

[7]Brown, S. A., "Displacement Extrapolations for CFD+CSM Aeroelastic Analysis," AIAA Paper 97-1090, Apr. 1997.

[8] Cebral, J. R., and Löhner, R., "Fluid-Structure Coupling: Extensions and Improvements," AIAA Paper 97-0858, Jan. 1997.

[9]Smith, M. J., Hodges, D. H., and Cesnik, C. E. S., "An Evaluation of Computational Algorithms to Interface Between CFD and CSD Methodologies," Wright-Patterson Laboratory, WL-TR-96-3055, Nov. 1995.

[10]Clutter, E. G., "A NURBS Based Interface Definition for Fluid-Structure Interactions Studies," Master's thesis, Department of Aerospace Engineering, Mississippi State University, Mississippi, Dec. 1997.

[11]Send, Wolfgang, "Coupling of Fluid and Structure for Transport Aircraft Wings," International Forum on Aeroelasticity and Structural Dynamics, CEAS/AIAA/ICASE/NASA Langley, Williamsburg, VA, June, 1999. (Not included in the bounded volume)

[12]Samareh, J. A., "Use Of CAD Geometry in MDO," The 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA-96-3991, Bellevue, WA, Sept. 1996, pp. 88–98.

[13]Murti, V., and Vallippan, S., "Numerical Inverse Isoparametric Mapping in Remeshing and Nodal Quantity Contouring," *Computers and Structures*, Vol. 22, No. 6, 1986, pp. 1011-1021.

[14]Pidaparti, R. M. V., "Structural and Aerodynamic Data Transformation Using Inverse Isoparametric Mapping," *Journal of Aircraft*, Vol. 29, No. 3, 1992, pp. 507-509.

[15]Maman, N., and Farhat, C., "Matching Fluid and Structure Meshes for Aeroelastic Computations: A Parallel Approach," *Computers and Structures*, Vol. 54, No. 4, 1995, pp. 779–785.

[16]Farhat, C., Lesoinne, M., and LeTallec, P., "Load and Motion Transfer Algorithms for Fluid/Structure Interaction Problems with Non-Matching Discrete Interface: Momentum and Energy Conservation, Optimal Discretization and Application to Aeroelasticity," *Computer Methods and Applied Mechanical Engineering*, Vol. 157, No. 1, 1998, pp. 95–114.

[17]Cebral, J. R., and Löhner, R., "Conservative Load Projection and Tracking for Fluid-Structure Problems," *AIAA Journal*, Vol. 34, No. 4, 1997, pp. 68-692.

[18]Cook, R. D., Malkus, D. S., and Plesha, M. E., *Concepts and Applications of Finite Element Analysis*, 3rd ed., John Wiley & Sons, New York, 1989.

[19]Samet, H., *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[20]Bonet, J. and Peraire, J., "An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems," *Int. J. Numer. Meth. Engng*, vol. 31, pp. 1-17, 1991.