Conservative Fitting for Multi-Disciplinary Analysis

John F. Dannenhoffer, III*
Aerospace Computational Methods Laboratory
Syracuse University, Syracuse, New York, 13244

Robert Haimes†
Aerospace Computational Design Laboratory
Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139

There has been a tendency in the last decade to transition from single-discipline to multi-disciplinary analysis and design; this trend has been hastened by the increased availability of analysis and optimization frameworks that manage the workflow. But one of the enabling technologies that is needed has not received much attention: that is, the need to transfer information from the surface of one representation to another, with particular emphasis on transferring the data conservatively. Described herein is a new method for performing such transfers. It is based upon two key technologies: a universal view of solver discretizations and a new optimization-enabled conservative fitting technique. The scheme is fully described and demonstrated in one dimension; then the extension to two dimensions is discussed along with results.

I. Introduction

In a Multi-Disciplinary Analysis and Optimization (MDAO) environment, there is the need to transfer data from one "surface" representation to another. These two representations can either be both from the same discipline but at different fidelities, or from different disciplines. One may want to transfer the "pressure" data from a structured quadrilateral grid used for Computational Fluid Dynamics (CFD) to the unstructured grid of mixed quadrilaterals and triangles used for Finite Element Analysis (FEA) structural calculations. Similarly, one may wish to transfer the displacements from the FEA mesh back to the CFD mesh.

While simple interpolation from one grid to the other is reasonably straightforward, it suffers from several problems:

- if one interpolates from surface A to surface B, and then back to surface A, there is a tendency for the interpolation errors to accumulate (that is, the original A and interpolated A may be quite different, especially if A and B have very different resolutions); and
- the interpolations tend not to be "conservative" (that is the integrated load on A does not in general match the integrated load seen on B).

This latter point is particularly acute when one is transferring variables such as pressure; without "conservative" fitting, the total load (lift) on the CFD and FEA meshes may be quite different. When used in an inner iteration (such as in weakly coupled fluid-structure interaction), any lack of conservation makes it

*Associate Professor, Mechanical and Aerospace Engineering, AIAA Associate Fellow.
†Principal Research Engineer, Department of Aeronautics & Astronautics, AIAA Member.

difficult to converge the procedure. It must be noted that conservation is a statement of "integration" and the integration needs to be consistent with the solver's internal assumptions (for conservation to hold).

To compound the above problems, it is often the case that the variables to be transferred can be represented in different ways on the two meshes. For example, it is not uncommon for the CFD to assume a piecewise linear, but continuous, representation within each element whereas the FEA may represent the quantity as piecewise constant. Any generally useful fitting scheme must be able to account for such differences in representations.

One of the earliest attempts at this coupling was made by Farhet, *et al.*;[1] this was later improved by Samareh.[2] In both cases, the values in the target distribution were computed so as to match the loads locally, with no attempt to ensure that the global load (and moments) were matched.

In order to circumvent these problems, a new conservative fitting procedure is developed here, which is based upon two key technologies:

- a unified view of solver discretizations that allows a user to define exactly how a function is represented within each element of a mesh; and
- a global "fitting" function that balances the need for quality interpolations and the need to ensure that the integrated quantities (i.e., loads) match on the two representations.

In the sections that follow, the conservative fitting is first described in one dimension. Then a section describes a new universal view of solver discretizations that has been developed. Finally these two technologies are combined into a two-dimensional conservative fitting scheme. The latter has been fully implemented for transfer between surfaces and is being integrated into the OpenMDAO environment.[3]

II. Conservative Fitting in 1D

In this section, the basic tools and techniques that will be needed in two dimensions are first developed and tested in one dimension. This is done to help explain the concepts in a setting where the visualization of results is simpler.

A. Discretizations

Consider a situation for which one wants to represent some function, f, using two different discretizations. We will call one of the discretizations the source, S, on which we already know the function values; the independent variables in this case are $u_{i,S}$ for $i = 1, \ldots, I_S$. The other discretization will be called the target, T, for which we want to find the dependent variables that best "fit" the source distribution; the independent variables here are $u_{i,T}$ for $i = 1, \ldots, I_T$. In general the domains will have the same extents, so that $u_{1,S} = u_{1,T}$ and $u_{I_S,S} = u_{I_T,T}$.

For each of the discretizations (S and T), the basis functions that are used to represent the functions can take a variety of forms:

- **piecewise constant - discontinuous**. Here the function can be represented as

$$\hat{f} = f_i \tag{1}$$

  for $u_i \leq u < u_{i+1}$. For the whole domain, there are $I - 1$ dependent variables, that is f_i for $i = 1, \ldots, I - 1$.

- **piecewise linear - continuous**. Here the function can be represented as:

$$\hat{f} = \left(\frac{u_{i+1} - u}{u_{i+1} - u_i} \right) f_i + \left(\frac{u - u_i}{u_{i+1} - u_i} \right) f_{i+1} \tag{2}$$

  for $u_i \leq u < u_{i+1}$. For the whole domain, there are I dependent variables, that is f_i for $i = 1, \ldots, I$.

- **piecewise linear - discontinuous**. Here the function can be represented as:

$$\hat{f} = \left(\frac{u_{i+1} - u}{u_{i+1} - u_i}\right) fl_i + \left(\frac{u - u_i}{u_{i+1} - u_i}\right) fr_i \tag{3}$$

  for $u_i \leq u < u_{i+1}$. For the whole domains, there are $2(I-1)$ dependent variables, that is fl_i and fr_i for $i = 1, \ldots, I-1$.

Other higher-order basis functions are also possible, but not implemented and presented here in one dimension.

B. Problem Formulation

As mentioned above, the best "fit" function is the one that simultaneously ensures that (1) the integral of the target distribution $\mathcal{I}_T \equiv \int f_T(u_T)\, du_T$ be the "same" as the integral of the source distribution, $\mathcal{I}_S \equiv \int f_s(u_S)\, du_S$, and that (2) the target distribution, $f_T(u_T)$ be "close" to the source distribution $f_S(u_S)$.

The integrals are computed cell by cell in a straightforward way, depending on the discretization type:

- **piecewise constant - discontinuous**

$$\mathcal{I} = \sum_{i=1}^{I-1} f_i (u_{i+1} - u_i) \tag{4}$$

- **piecewise linear - continuous**

$$\mathcal{I} = \sum_{i=1}^{I-1} \frac{f_i + f_{i+1}}{2} (u_{i+1} - u_i) \tag{5}$$

- **piecewise linear - discontinuous**

$$\mathcal{I} = \sum_{i=1}^{I-1} \frac{fl_i + fr_i}{2} (u_{i+1} - u_i) \tag{6}$$

The first difficulty in the above is defining a quantitative means of measuring how "close" two distributions are to each other. This is particularly difficult if one considers combinations of representations such as piecewise constant (discontinuous) and piecewise linear (continuous). To solve this problem, define a series of "match points" at which we want the representations to match. This idea was inspired by the concept of "quadrature points" in Gaussian quadrature. The writer of each representation has total freedom in choosing the "match points" that best describe the closeness of fit. For the discretizations described above:

- **piecewise constant - discontinuous**. Here there are two match points per segment, defined as $\tilde{u}_{2i-1} = (3u_i + u_{i+1})/4$ and $\tilde{u}_{2i} = (u_i + 3u_{i+1})/4$. (This choice was somewhat arbitrary, but was chosen so that the match points were evenly distributed in the domain.) Hence the number of match points is $n_m = 2(I-1)$ in these examples.

- **piecewise linear - continuous**. Here the match points are chosen to lie at the independent variable locations $\tilde{u}_i = u_i$. Here the number of match points is $n_m = I$

- **piecewise linear - discontinuous**. As above, there are two match points per segment, defined as $\tilde{u}_{2i-1} = (3u_i + u_{i+1})/4$ and $\tilde{u}_{2i} = (u_i + 3u_{i+1})/4$. Here there are $n_m = 2(I-1)$ match points.

The second difficulty is in balancing the fact that the integrals should match and that the functions be "close" to each other. To do this, a composite objective function is defined as

$$\begin{aligned} \mathcal{J} &= W_A \left(\mathcal{I}_S - \mathcal{I}_T\right)^2 \\ &+ W_S \sum_{j=1}^{n_{m,S}} \left(\hat{f}_S(\tilde{u}_{j,S}) - \hat{f}_T(\tilde{u}_{j,S})\right)^2 \\ &+ W_T \sum_{j=1}^{n_{m,T}} \left(\hat{f}_S(\tilde{u}_{j,T}) - \hat{f}_T(\tilde{u}_{j,T})\right)^2 \end{aligned} \tag{7}$$

Here, W_A, W_S, and W_T are user-selectable weight factors used to define the relative importance of the three terms. The term on the first line ensures that the integrals match; a large value of W_A causes the interpolation to be conservative. The term on the second line causes the interpolated value in the source $\hat{f}_S$ to match the interpolated value in the target $\hat{f}_T$ at the source match points $\tilde{u}_{j,S}$ for $j = 1, \ldots, n_{m,S}$. The term on the third line causes the interpolated value in the source $\hat{f}_S$ to match the interpolated value in the target $\hat{f}_T$ at the target match points $\tilde{u}_{j,T}$ for $j = 1, \ldots, n_{m,T}$.

An optimization problem can thus be formed as: find the $f_{i,T}$ that minimizes $\mathcal{J}$. This problem can easily be solved by a conjugate-gradient optimizer.[4] To compute the gradient needed in the optimizer, one can resort to finite differences if the number of design variables is small, as it is in most one-dimensional problems. However when implemented in two dimensions (see Section IV), the number of optimization design variables can be quite large, making finite-difference gradient calculations prohibitively expensive. To circumvent this, an adjoint version of the objective function is formed and used.

Define $\bar{f}_{i,T} \equiv \left(\nabla_{f_{i,T}}\right) \mathcal{J}$ as the gradient of the composite objective function $\mathcal{J}$ with respect to the design variables $f_{i,T}$. This can be computed by "backwards differentiating" the function that computes $\mathcal{J}$ either via a automatic code differentiator (such as Tapenade[5]) or by hand; the latter approach is used here.

As an example, consider the computation of the "conservative" part of the composite objective function where the source is assumed to be piecewise constant and the target is assumed to be piecewise linear - continuous:

$$\begin{aligned} \mathcal{I}_s &= \sum_{i=1}^{I_S-1} f_{i,S}(u_{i+1,S} - u_{i,S}) \\ \mathcal{I}_T &= \sum_{i=1}^{I_T-1} \frac{f_{i,T} + f_{i+1,T}}{2}(u_{i+1,T} - u_{i,T}) \\ \mathcal{X} &= W_A \left(\mathcal{I}_S - \mathcal{I}_T\right)^2 \end{aligned}$$

The "backward code" runs these statements in the reverse order and each statement backwards. First the derivatives are initialized:

$$\begin{aligned} \bar{\mathcal{X}} &= 1 \\ \bar{f}_{i,T} &= 0 \quad \text{for} \quad i = I_T - 1, I_T - 2, \ldots, 1 \end{aligned}$$

Next, the equation for $\mathcal{X}$ is executed backwards:

$$\begin{aligned} \bar{\mathcal{I}}_S &= +2W_A(\mathcal{I}_s - \mathcal{I}_T)\bar{\mathcal{X}} \\ \bar{\mathcal{I}}_T &= -2W_A(\mathcal{I}_s - \mathcal{I}_T)\bar{\mathcal{X}} \end{aligned}$$

And then for $i = I_T - 1, I_T - 2, \ldots, 1$ the equation for $\mathcal{I}_T$ is executed backwards:

$$\begin{aligned} \bar{f}_{i,T} &= \bar{f}_{i,T} + \frac{u_{i+1,T} - u_{i,T}}{2}\bar{\mathcal{I}}_T \\ \bar{f}_{i+1,T} &= \bar{f}_{i+1,T} + \frac{u_{i+1,T} - u_{i,T}}{2}\bar{\mathcal{I}}_T \end{aligned}$$

Note that there is no backwards calculation for $\mathcal{I}_S$ since it is not a function of the dependent variables ($f_{i,T}$).

Similar operations are used for all other terms in the equations. This latter technique has been found to reduce the computational effort to scale linearly with the number of unknowns (as opposed to quadratically without the adjoints).

C. One-dimensional Examples

To demonstrate the conservative fitting scheme in one dimension, a series of test cases were executed with one of three source and target discretizations:

A a continuous linear discretization on 32 uniformly-spaced points;

B a discontinuous linear discretization on 8 randomly-spaced points; and

C a (discontinuous) constant discretization on 16 randomly-spaced points.

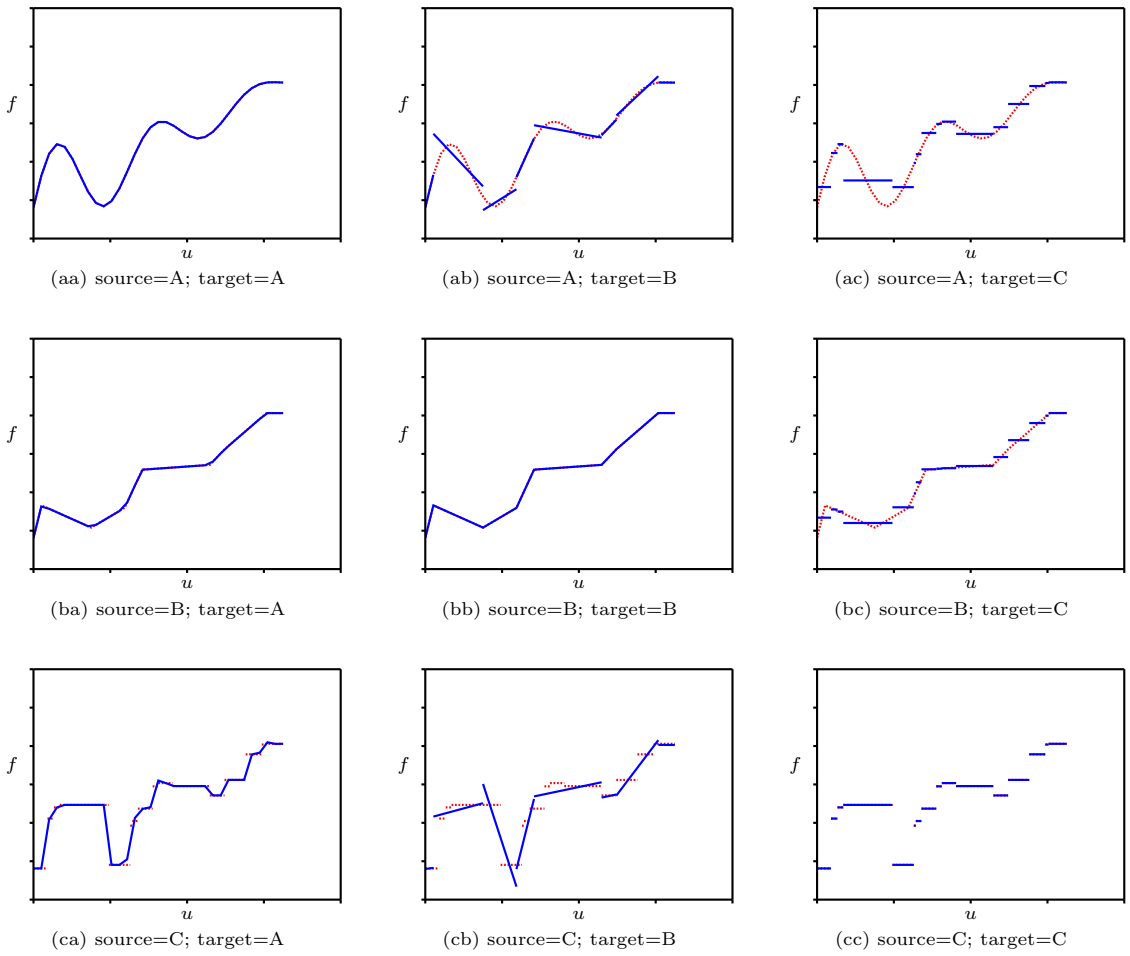


Figure 1. Sample one-dimensional results for $W_A = 10^6$, $W_S = W_T = 1$. Case A has 32 uniformly-spaced points and a linear, continuous discretization; case B has 8 randomly-spaced points and a linear, discontinuous discretization; case C has 16 randomly-spaced points and a constant, discontinuous discretization. Source distributions are shown by dotted red lines and target distributions are shown by solid blue lines.

The result of these cases is shown in Figure 1; in all cases $W_A = 10^6$ and $W_S = W_T = 1$ (which requires that the integrals match almost exactly). In the cases where the source and target match, one cannot see the source (dotted red line) since it lies directly behind the target (solid blue line) distribution.

The cases for "target=B" shows the effect of the match points being at the one- and the three-quarter points in each target (blue) segment. One can see the over-shoots that result; this will be particularly important in analyzing the two-dimensional results.

Table 1. Sample one-dimensional results for $W_A = 10^6$, $W_S = W_T = 1$. Case A has 32 uniformly-spaced points and a linear, continuous discretization; case B has 8 randomly-spaced points and a linear, discontinuous discretization; case C has 16 randomly-spaced points and a constant, discontinuous discretization.

	Source Case	Target Case		
		A	B	C
E_A	A	.0000000	.0000002	.0000002
E_M		.0000000	.0067688	.0071512
RMS_S		.0000000	.0803391	.1824280
RMS_T		.0000000	.0283419	.0380904
E_A	B	.0000000	.0000000	.0000000
E_M		.0027555	.0000000	.0010031
RMS_S		.0013490	.0000000	.0712629
RMS_T		.0009638	.0000000	.0374163
E_A	C	.0000000	.0000003	.0000000
E_M		.0035372	.0043249	.0000000
RMS_S		.0532094	.1080616	.0000000
RMS_T		.0206119	.0776208	.0000000

The errors associated with each of these cases are given in Table 1. In the table E_A is the absolute values of the difference in the areas $|\mathcal{I}_S - \mathcal{I}_T|$, E_M is the absolute value of the difference in the first moments $\left|\int_S f u\, du - \int_T f u\, du\right|$, RMS_S is the root-mean-square error at the source match points, and RMS_T is the root mean square at the target match points. Not surprisingly, each of these errors is exactly zero for the cases where the source and target distributions match. For the other cases, one can see that the area errors E_A are quite low; in fact, these can be made as small as desired by increasing the value of W_A (but at the expense of more steps required in the conjugate-gradient optimizer). Note however that the values of E_M are not very small, owing to the fact that the optimization did not explicitly try to drive this to zero. If another term, such as $W_M \left(\int_S f u\, du - \int_T f u\, du\right)^2$ were added to Eq. 7, then this term could also be driven to essentially zero.

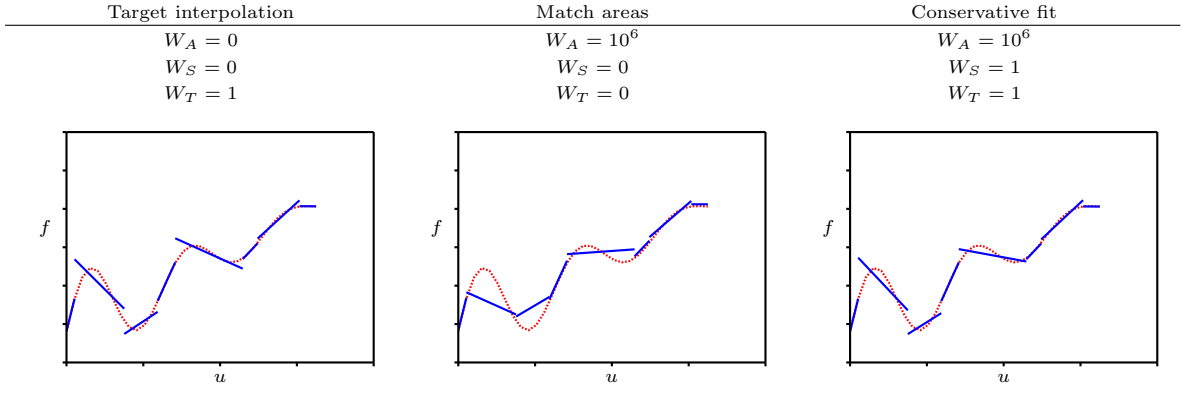


Figure 2. Sample one-dimensional results for source = A (32 uniformly-spaced points and a linear, continuous discretization) in dotted red and target = B (8 randomly-spaced points and a linear, discontinuous discretization) in solid blue.

To determine the effect of enforcing (to some extent) conservativeness, each of these cases was re-executed with differing values of the penalty function weights. While the results for all cases are similar; the detailed results for the case where A (linear continuous discretization on 32 uniformly-spaced points) is the source and B (linear discontinuous discretization on 8 randomly-spaced points) is shown in Figure 2 and the errors are tabulated in Table 2. The first column shows the effect of the traditional method of transferring data, that is "target interpolation", which is achieved by setting $W_A = W_S = 0$ and $W_T = 1$. The fit is as expected and the errors show quite a large "area error", a moderate "source error", and no "target error".

Table 2. Sample one-dimensional results for source = A (32 uniformly-spaced points and a linear, continuous discretization) and target = B (8 randomly-spaced points and a linear, discontinuous discretization).

	Target interpolation	Match areas	Conservative fit
W_A	0	10^6	10^6
W_S	0	0	1
W_T	1	0	1
E_A	.0108619	.0000000	.0000002
E_M	.0054605	.0211274	.0067688
RMS_S	.0976927	.1916839	.0803391
RMS_T	.0000000	.1579835	.0283419
$D_{A,25}$	.0008840	.0000000	.0000001

The second column shows the effect of simply matching the areas, with no regard for how "close" the source and target distributions are to each other; this is achieved by setting $W_A = 10^6$ and $W_S = W_T = 0$. As expected, the "area error" vanishes, but the moment and matching errors are quite large.

Finally the last column shows the effect of the "conservative fit", which is achieved by setting $W_A = 10^6$ and $W_S = W_T = 1$. Here the "area error" is quite small (especially as compared with the error in the first column) and the "matching errors" are also quite small. Since the moment was not considered here as part of the optimization, the values of E_M are rather large for all three cases.

Table 2 also contains a row entitled $D_{A,25}$, which is the *drift* or accumulated errors in the source area after 25 cycles of transferring the source to the target and back to the source. As can be seen, the "match areas" and "conservative fit" schemes show no accumulated error, whereas the "target interpolation" method shows a rather substantial accumulation; this error accumulation can have detrimental effects if the "target interpolation" scheme is used in fluid/structure coupling in an MDAO environment.

III. Universal View of Solver Discretizations

In examining the procedure performed above, the only operations that are required on any discretization are:

- **Interpolation**. Given some location with an element (typically defined in terms of a barycentric coordinate s) and the values of the dependent variables that support the element, determine the interpolated value of the dependent variable(s).
- **Inverse interpolation**. Given some dependent variable(s) within an element (for example x), determine the barycentric coordinate(s) (s) that correspond with the specified dependent variables.
- **Integration**. Given some element and the values of the dependent variables that support the element, find the integral of the dependent variable over the element.

- **Definition of match points.** Given an element, define the locations of the "match points" with the element at which one wants to measure the "closeness" of the fit.

Of course to be useful in the conjugate-gradient optimizer, the adjoint version of the interpolation and integration functions are also needed. This means that for a general MDAO framework, the functions described above are required as well as the data that supports the interpolation/integration within a discipline. This can cause the following problems for the framework if it is responsible for the implementation of the above functions:

- **Consistency**. What ensures that the formulation assumed by the framework is the same as the one used by the solver? This is particularly acute in the cases where the form of interpolation is not rigorously defined such as with continuous *cell-centered* data.
- **Full breath of support**. There is a fairly limited suite of discretizations used in finite volume formulations, but within the realm of finite element methods there are multiple variations in basis functions (polynomial, spectral and now NURBS) as well as the *order* of the approximation. This would put an extraordinary burden on the framework itself and it would be the responsibility of the framework developers to add the appropriate functionality.

These problems are mitigated by using a plugin scheme within the framework to dynamically load, at run time, the solver specific interpolation/integration functionality. This code must define specific *entry points* that provide the above as well as a function that returns the organization of the boundary (surface) data to the framework. This has the advantage of only requiring the framework to capture and transmit the data used in conservative fitting (and not the details of how the interpolation/integration needs to be performed). But for this to work it is important to define a universal view of boundary-based spatial discretizations (the 2D analogue of what has been described in the previous section) so that the data can be properly interrogated by the framework and the plugin functions.

A. General Element Data Structure

A review of the literature finds nothing that attempts to generalize, unify and codify the kinds of spatial discretizations found within AMR (embedded boundary), finite volume, finite difference and finite element methods. An obvious candidate would be CGNS,[6] but a careful examination of the data structures reveals native support for only simple finite volume and finite difference schemes.

Described below is the element data structure developed that allows one to represent any discretization of which the authors are aware. This structure defines point positions in a *facet* reference space (defined using the 2 coordinates $[s, t]$) and has simple rules that hold the information together. Collections of the element definitions can exist for any *boundary*.

This general 2D element data structure is driven by these sizes:

- **nGeom** is the number of positions (per cell) at which the physical extent of the element is defined.
- **nData** is the number of positions used to define the data locations in the element. A 0 indicates that geometric and data positions are equivalent.
- **nSide** the number of sides of the polygon (which also defines the number of neighboring elements). Note that higher-order positions (along the sides) do not contribute to the polygonal shape.

The fundamental numbering spaces are:

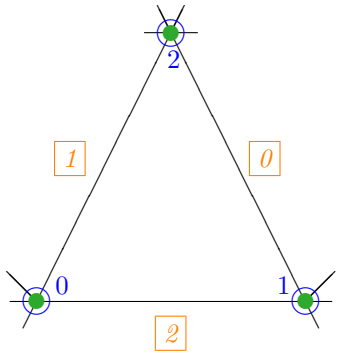
- **Geometry reference points** that contain the barycentric coordinates (in the *facet* reference space) at which the independent variables are defined. In many cases this will correspond to the corners of the element. The ordering of geometric points should first contain those positions that are the corners of the elements and then any ancillary positions and should be done using a right-handed traversal. This facilitates a simple scheme for neighbor/side numbering.

- **Data reference points** that contain the barycentric coordinates at which the dependent variables are defined. nData of zero indicates that the data reference points coincide with the geometry reference points (as it will for piecewise-bilinear continuous discretizations). In some cases, separating the geometry and data reference points is essential for higher-order representations, such as are used in discontinuous-Galerkin methods.
- **Neighbors** contain element indices that refer to the neighboring elements (nSide in length). *Side 0* always refers to the segment containing *geometry position 1 & 2* and the numbering again follows a right handed scheme.

Each element of a type as defined by this structure contains indices into a list of vertices. Discontinuous data (at element bounds) has differing indices on either side of the element interface.

Figures 3 through 7 display various pictorial examples of specifying the spatial discretization for differing schemes within this unifying data architecture:

nGeom = 3, nData = 0, nSide = 3



Barycentric Coordinates

geom&data	s	t
0	0	0
1	1	0
2	0	1

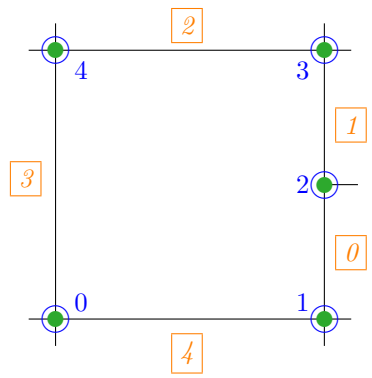
Neighbor Cells

side	segment	
0	1	2
1	2	0
2	0	1

Figure 3. Simple continuous bilinear triangle.

Figure 3 shows an example of a simple continuous triangle where bilinear interpolation is used to specify data within the element. This is the kind of triangle you would get from a simple finite volume scheme or a first-order polynomial finite element code. In this case the *geometric* positions coincide with the *data* positions. The element simply has 3 corner positions and 3 sides.

nGeom = 5, nData = 0, nSide = 5



Barycentric Coordinates

geom&data	s	t
0	0	0
1	1	0
2	1	1/2
3	1	1
4	0	1

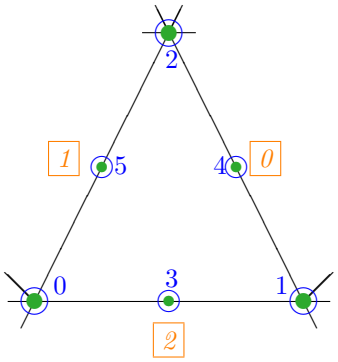
Neighbor Cells

side	segment	
0	1	2
1	2	3
2	3	4
3	4	0
4	0	1

Figure 4. Hanging vertex (AMR) quadrilateral.

Figure 4 displays a hanging-node along a single mid-side of a quadrilateral that you would encounter in an AMR-style mesh. Again, nData is zero meaning that the *data* positions are the same as the vertices that support the element (the *geometric* positions). In this case there are 4 corners, but because of the mid-side split there are 5 vertices and 5 sides. Note the right-handed positioning and side numbering.

nGeom = 6, nData = 0, nSide = 3



Barycentric Coordinates

geom&data	s	t
0	0	0
1	1	0
2	0	1
3	1/2	0
4	1/2	1/2
5	1/2	0

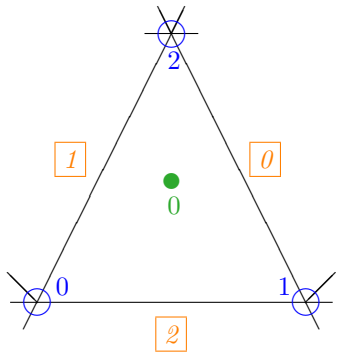
Neighbor Cells

side	segment	
0	1	2
1	2	0
2	0	1

Figure 5. Second order continuous triangle.

Figure 5 shows another discretization that does not have unique *data* positions. This is an example of a nodal second-order-polynomial triangle. Note that the corner positions and the neighboring information are the same as seen for the simple bilinear triangle (Figure 3). The higher-order *geometric positions* (4, 5 & 6) are not used in determining the facet sides and in this case nGeom is not equal to nSide.

nGeom = 3, nData = 1, nSide = 3



Continuous/discontinuous depends
on interpolation scheme used

Barycentric Coordinates

geom	s	t
0	0	0
1	1	0
2	0	1
data		
0	1/3	1/3

Neighbor Cells

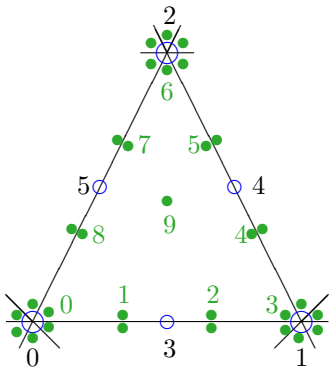
side	segment	
0	1	2
1	2	0
2	0	1

Figure 6. Cell-centered or discontinuous constant triangle.

Figure 6 shows a case where the data values are stored a locations different than the positions used to define the *corners* of the element. In all respects this is the same as seen in Figure 3 except that there is a single data value located at the centroid of the element. This can reflect a constant discontinuous triangle or continuous data depending on the interpolation applied over the element.

nGeom = 6, nData = 9, nSide = 3

Neighbors same as second-order
continuous triangle



Barycentric Coordinates

geom	s	t
0	0	0
1	1	0
2	0	1
3	1/2	0
4	1/2	1/2
5	1/2	0

data	s	t	data	s	t
0	0	0	5	1/3	2/3
1	1/3	0	6	0	1
2	2/3	0	7	0	2/3
3	1	0	8	0	1/3
4	2/3	1/3	9	1/3	1/3

Figure 7. Discontinuous triangle (q=2, p=3).

Figure 7 displays the mapping for a second order geometric representation with a third-order discontinuous storage scheme. In this case the *geom* positions and the shape of the element is the same as seen in Figure 5, so the neighboring information is also identical. The multiple vertices seen for all *data* positions (except for 9) indicates that there are multiple data values coincident to that position (each contained in this and the neighboring elements).

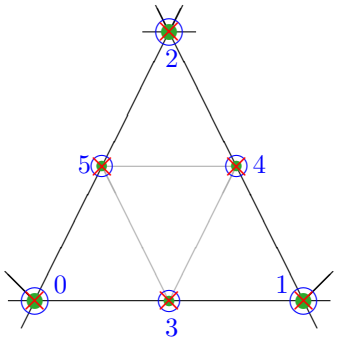
B. Element Data Structure for Conservative Transfers

The conservative fitting scheme requires ancillary data to to be added to the unified view of discretizations as described above. This additional data facilitates the tasks of locating positions in another discipline's boundary (discrete surface representation) and matching the values at those locations. This adds the following to the universal discretization structure:

- **nMat** is the number of points in the element used to define interpolation matching within the element. A 0 indicates that geometric and match positions are the same.
- **nTris** is the number of triangles that the element should be broken up into so that an *in-triangle* predicate can be used to determine interior locations.
- **Match points** that contain the barycentric coordinates (in the reference *facet*) at which the "closeness" of the fit will be evaluated, as described in the 1D fitting. These match points can coincide with either the geometry or data reference locations or can be distinct from these (as was used in the piecewise-constant discontinuous representation described in the 1D section).
- **Triangles** allow the positions (in $[u, v]$ of the surface) within the element – if not a triangle) to be simply found. The indices should also be listed in a right-handed manner. This data is also useful for visualizing the discipline's bounding data.

It should be noted that the fitting scheme does not need element neighboring information so this is not shown in Figures 8 through 10.

nTris = 4, nGeom = 6
nData = 0, nMat = 0



Barycentric Coordinates

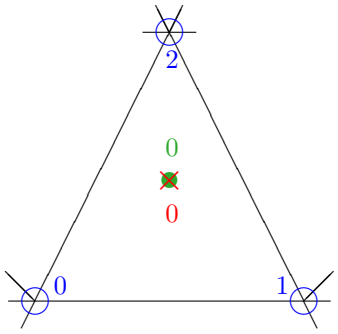
geom&data	s	t
0	0	0
1	1	0
2	0	1
3	1/2	0
4	1/2	1/2
5	1/2	0

Triangle Indices

0	3	5
3	1	4
5	4	2
3	4	5

Figure 8. Second order continuous triangle.

When comparing Figure 8 to Figure 5 one notices that much is exactly the same. Because the fitting will be continuous the match points are identical to the *geom* positions, the only additional data is the definition of the triangles used to tessellate the larger element.



nTris = 1, nGeom = 3
nData = 1, nMat = 1

Barycentric Coordinates

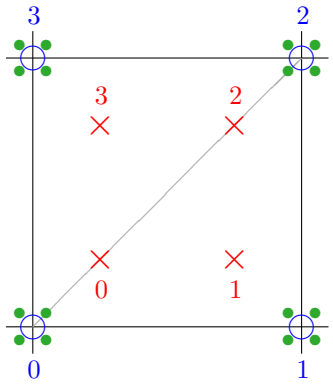
geom	s	t
0	0	0
1	1	0
2	0	1
data		
0	1/3	1/3
match		
0	1/3	1/3

Triangle Indices

0	1	2

Figure 9. Cell-centered or discontinuous constant triangle.

Figure 9 displays the same spatial discretization as seen in Figure 6. In this case, the location for matching is the position where the data is stored.



nTris = 2, nGeom = 4
nData = 4, nMat = 4

Barycentric Coordinates

geom	data	s	t
0	0	0	0
1	1	1	0
2	2	1	1
3	3	0	1
match			
0		1/4	1/4
1		3/4	1/4
2		3/4	3/4
3		1/4	3/4

Triangle Indices

0	1	2
0	2	3

Figure 10. Discontinuous bilinear quadrilateral.

Figure 10 depicts a discontinuous (bilinearly interpolated) quadrilateral. The quadrilateral is simply broken into 2 triangles for the $[u, v]$ locating function. The *data* and *geom* baycentric positions match up but need to be unique so that differing indices can populate the corners because multiple data values can exist (one per element touching the location). It should be noted that if the match positions were at the same locations as the *geometric* positions, then the resultant conservative data transfer would look continuous.

IV. Extension and Validation in 2D

A. Problem Formulation

The formulation of the problem in two dimensions is a straightforward extension of the one-dimensional scheme described in Section II. The chief differences are that in 2-D:

- there are two independent variables (u and v) that refer to the parametric position on the surface (note that both discretizations must refer to the same $[u, v]$ space);
- the functional forms are constant, bilinear or potentially high-order; resulting in expressions for $\hat{f}$ that are more complex;
- the expressions for the integrals $\mathcal{I}_S$ and $\mathcal{I}_T$ involve an area integral; and
- the "match points" are as described in Section III.

There are no other changes in going from one to two dimensions.

B. Validation Problem

The sample problem used to validate the new technique in two dimensions is a wing defined in four ways:

A a single surface representing the entire upper surface of the wing, which is discretized with a moderate number of triangles that support a bilinear-continuous approximation;

B two surfaces (one representing the inboard half and the other the outboard half) of the wing; there is a small gap (1% span) between the two sections. Each of these parts are discretized with quadrilaterals that support a bilinear-continuous approximation;

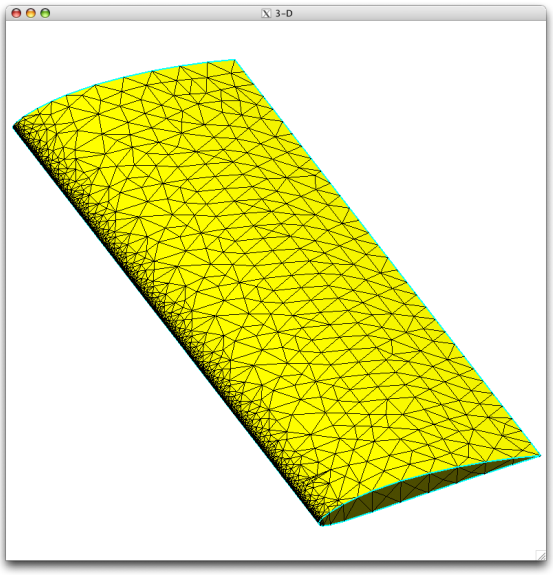
C five surfaces (two representing slats, one representing a flap, one representing an access cover, and the other the rest of the wing). There is a single set of triangles that cover these areas, which support a bilinear-discontinuous approximation; and

D a single surface, which is discretized with a very coarse set of triangles, which support a constant-discontinuous approximation.

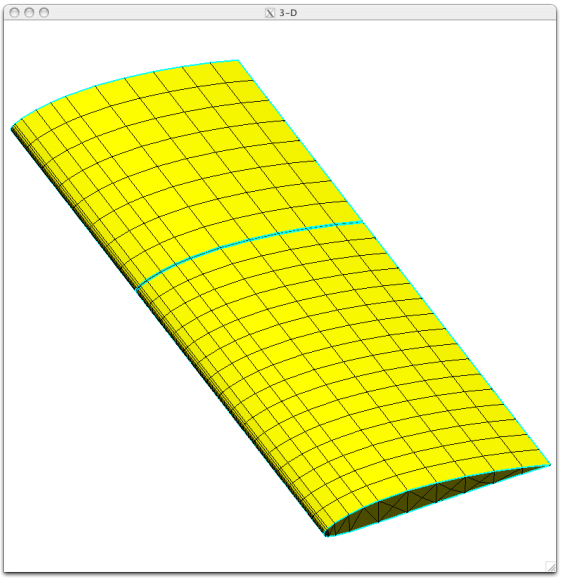
These discretizations are shown in Figure 11.

As was done in the one-dimensional examples, a function was generated on each of the source discretizations and the transferred to each of the target discretizations. For example in Figure 12, the function $f_s = x$ was defined on A (the triangles supporting a linear-continuous approximation). The results of applying the conservative fitter to each of the target discretizations is shown. Similar results are shown in Figure 13 where $f_s = z$ is defined on B, in Figure 14 where $f_s = x + z$ is defined on C, and in Figure 15 where $f_s = xz$ is defined on D.

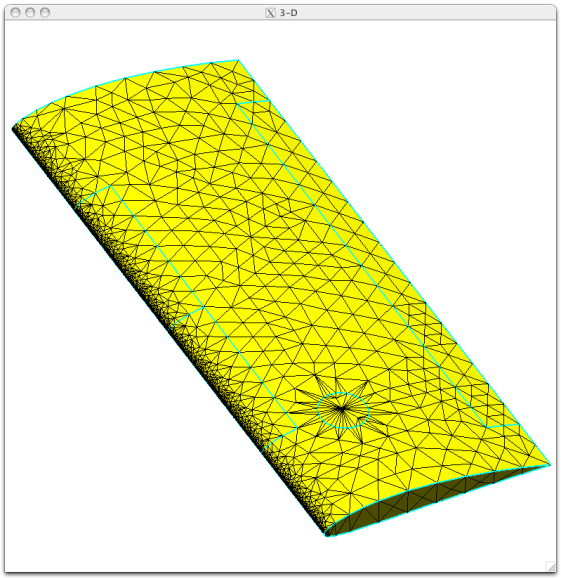
One area of note is the "red" region in Figure 15 where the target=C. This large local error is attributed to the "overshoot" that can occur when using a bilinear-discontinuous approximation, especially on coarse source data. This is seen in the 1D example depicted in Figure 1(cb), where the source is (discontinuous) constant and the target is a discontinuous linear discretization.



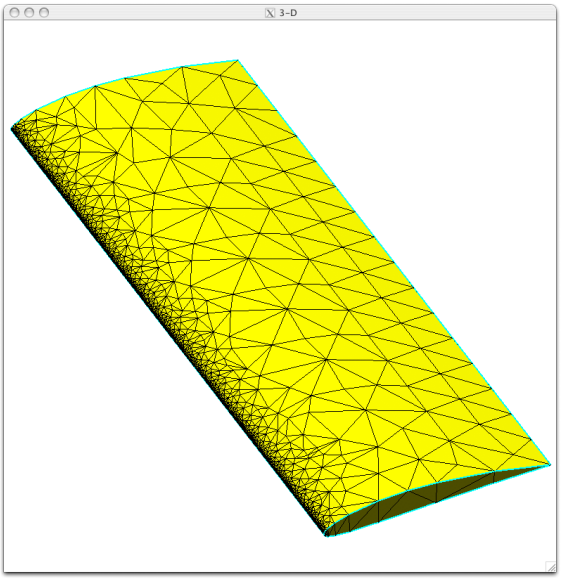
A: triangles supporting bilinear-continuous



B: quadrilaterals supporting bilinear-continuous

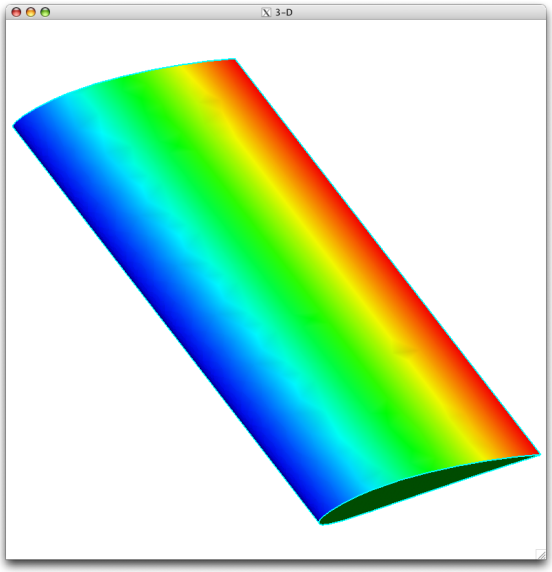


C: triangles supporting bilinear-discontinuous

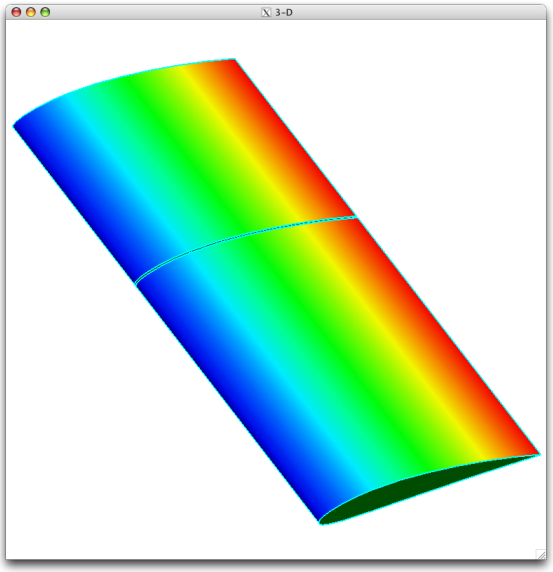


D: triangles supporting constant-discontinuous

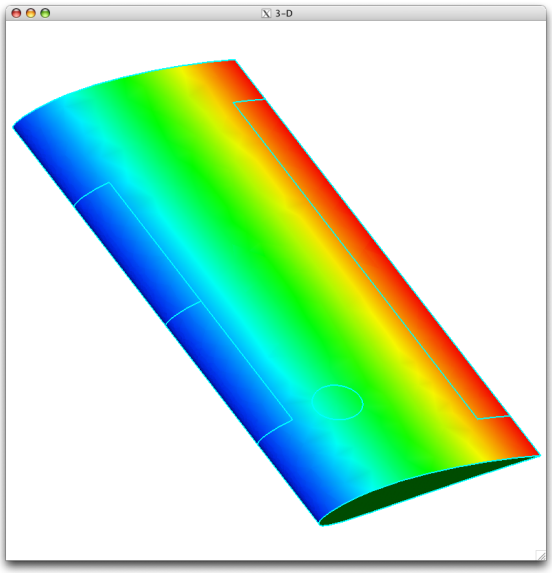
Figure 11. Four discretizations of the sample wing.



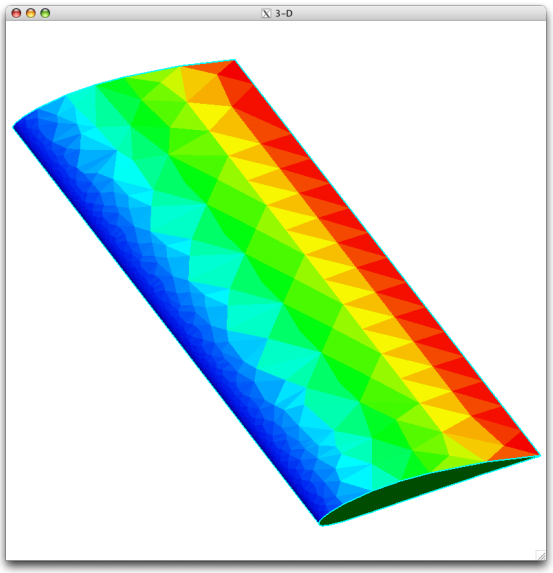
Target A: triangles supporting bilinear-continuous



Target B: quadrilaterals supporting bilinear-continuous

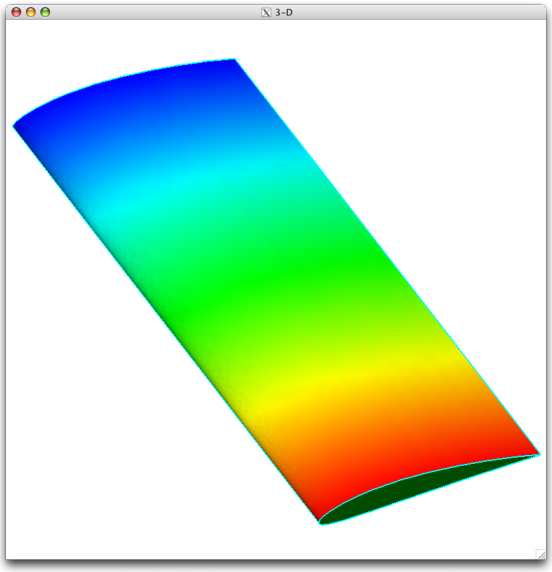


Target C: triangles supporting bilinear-discontinuous

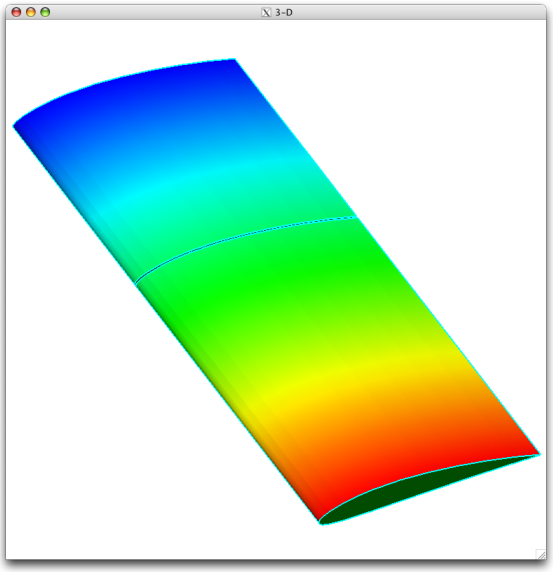


Target D: triangles supporting constant-discontinuous

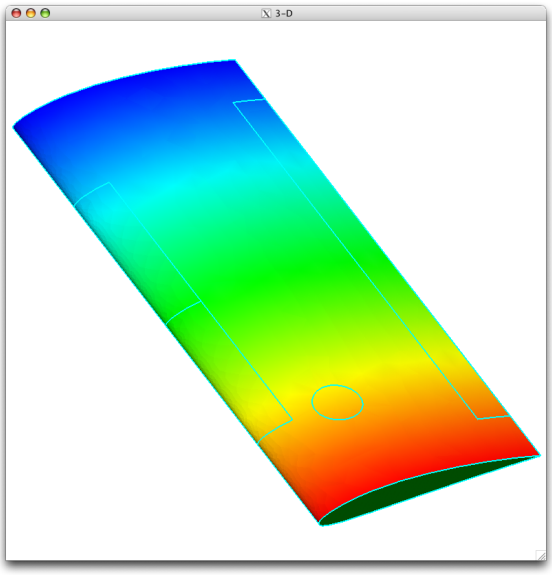
Figure 12. Results obtained on the four target discretizations with the function $f_S = x$ defined on the source A (triangular-bilinear-continuous discretization). $W_A = 10^6$ and $W_S = W_T = 1$.



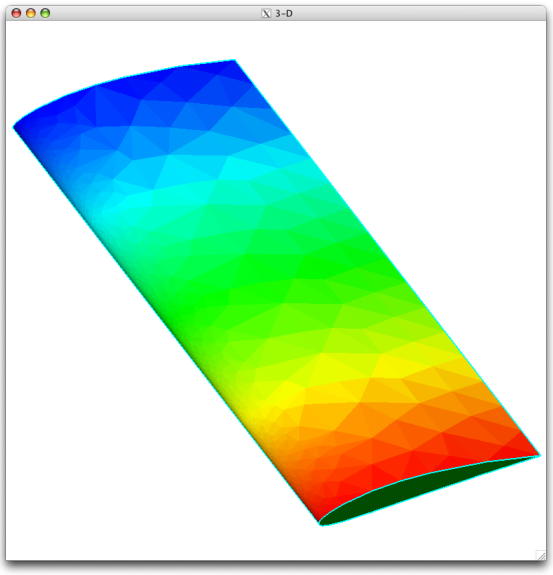
Target A: triangles supporting bilinear-continuous



Target B: quadrilaterals supporting bilinear-continuous

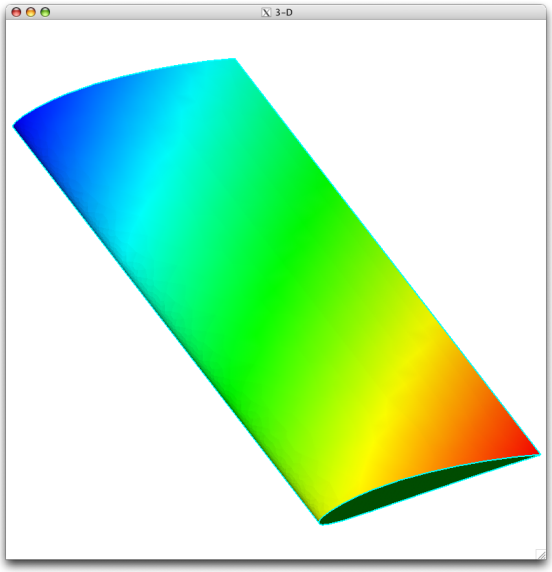


Target C: triangles supporting bilinear-discontinuous

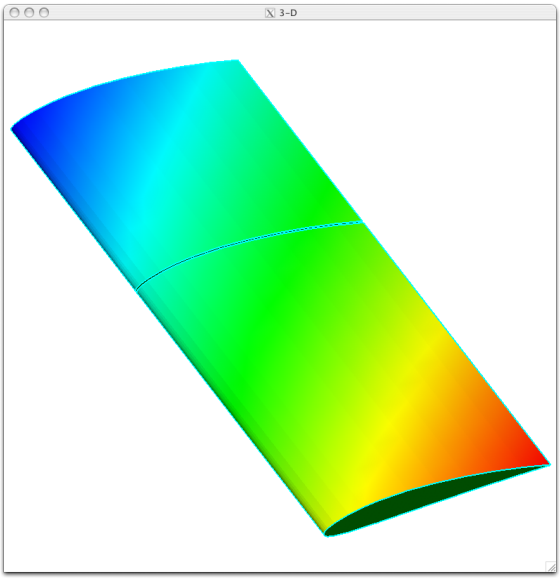


Target D: triangles supporting constant-discontinuous

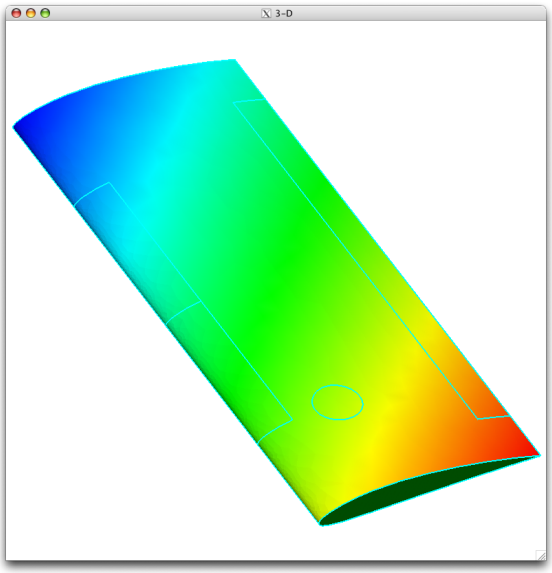
Figure 13. Results obtained on the four target discretizations with the function $f_S = z$ defined on the source B (quadrilaterals supporting bilinear-continuous discretization). $W_A = 10^6$ and $W_S = W_T = 1$.



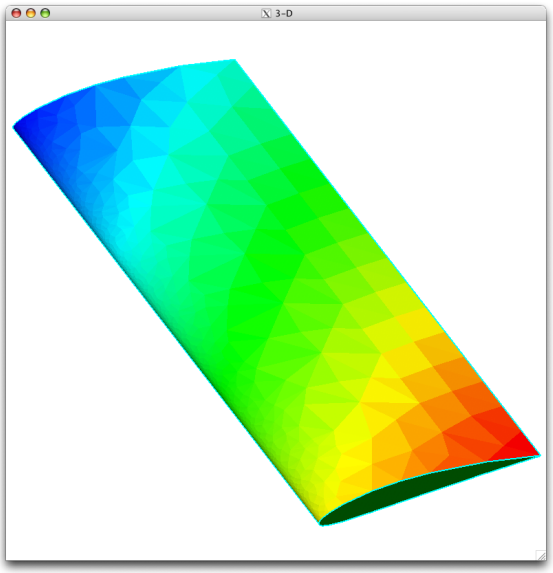
Target A: triangles supporting bilinear-continuous



Target B: quadrilaterals supporting bilinear-continuous

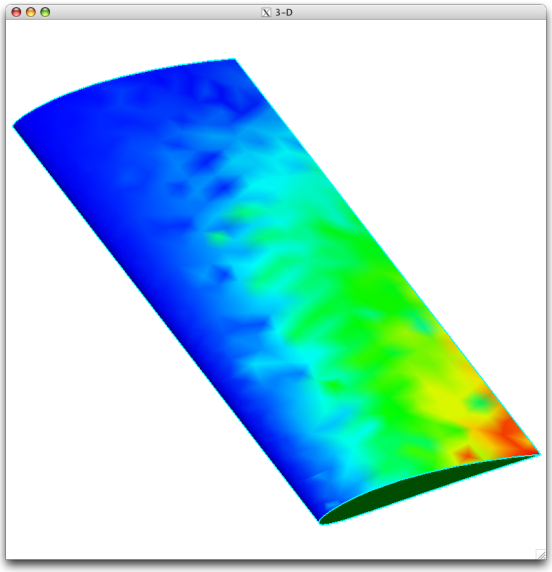


Target C: triangles supporting bilinear-discontinuous

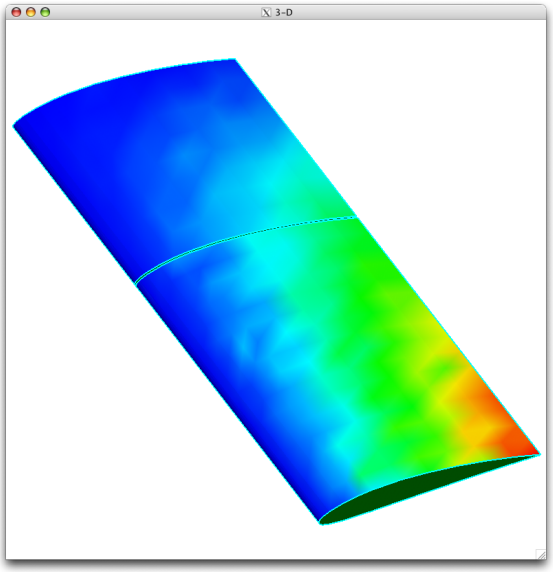


Target D: triangles supporting constant-discontinuous

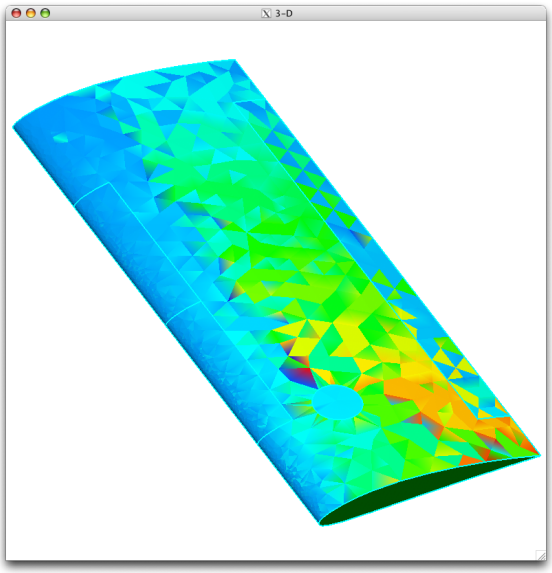
Figure 14. Results obtained on the four target discretizations with the function $f_S = x + z$ defined on the source C (triangles supporting bilinear-discontinuous discretization). $W_A = 10^6$ and $W_S = W_T = 1$.



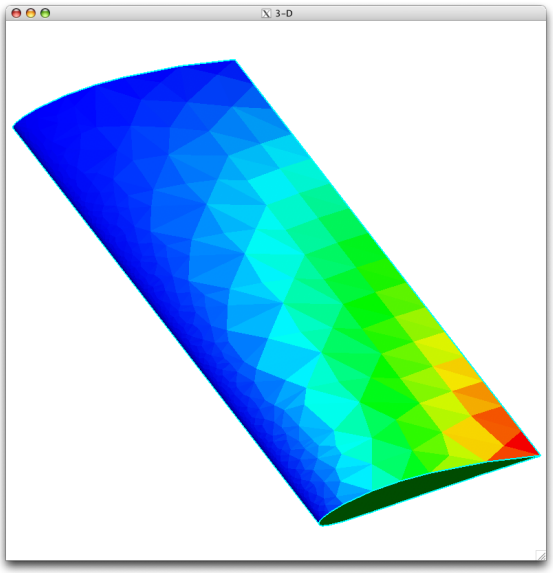
Target A: triangles supporting bilinear-continuous



Target B: quadrilaterals supporting bilinear-continuous



Target C: triangles supporting bilinear-discontinuous



Target D: triangles supporting constant-discontinuous

Figure 15. Results obtained on the four target discretizations with the function $f_S = xz$ defined on the source D (triangles supporting constant-discontinuous discretization). $W_A = 10^6$ and $W_S = W_T = 1$.

To compare the effectiveness of the current "conservative fitting" with that of traditional interpolations, the average interpolation errors and the area differences are shown in Table 3 for all 16 cases. In all cases, the area differences are quite high for interpolation, whereas the conservative fitting essentially drives the area difference to zero at the expense of a slight increase in average target error.

Table 3. Errors associated with transferring from each of the source to each of the possible target distributions.

	Source: A			
	Interpolation		Conservative fit	
Target	Avg target error	Area diff	Avg target error	Area diff
A	0	0	0	0
B	.000004	8.7e-2	.000884	3.2e-7
C	.000145	1.7e-4	.000146	6.3e-10
D	.039420	8.6e-1	.040162	3.2e-6

	Source: B			
	Interpolation		Conservative fit	
Target	Avg target error	Area diff	Avg target error	Area diff
A	.000422	5.2e-4	.002310	1.1e-8
B	.000006	8.7e-2	.000954	1.8e-6
C	0	0	0	0
D	.080817	2.7e-1	.200731	5.5e-6

	Source: C			
	Interpolation		Conservative fit	
Target	Avg target error	Area diff	Avg target error	Area diff
A	.000185	4.1e-2	.002339	9.4e-7
B	0	0	0	0
C	.000165	1.3e-1	.006960	2.9e-7
D	.111407	4.2e-1	.112723	9.6e-7

	Source: D			
	Interpolation		Conservative fit	
Target	Avg target error	Area diff	Avg target error	Area diff
A	.000279	1.6e-3	.000281	3.5e-9
B	.000017	8.3e-2	.000393	1.9e-7
C	.000247	5.4e-3	.000264	1.2e-8
D	0	0	0	0

V. Conclusions

A new method for performing conservative data transfers has been presented. It is based upon two key technologies: a universal view of solver discretizations and a new optimization-enabled conservative fitting technique. The scheme is fully described and demonstrated in one dimension; then the extension to two dimensions is discussed along with selected two-dimensional results.

This new technique can be used as the basis for the transfer of information between discretized surfaces in an MDAO setting. The various surfaces can come from different disciplines, or can come from a single discipline but at two different fidelities.

The described scheme is being implemented in the **G**eometry **E**nvironment for **M**DAO within the OpenMDAO framework.[3] Results to date show that this new technique can perform the necessary "fittings" quickly enough so it appears to be a relatively insignificant additional cost as compared to straight interpolation.

Acknowledgement

This work was funded by NASA Cooperative Agreement NNX11AI66A: Topic 2.1 - MDAO Open-source Engineering Framework; Chris Heath is the Technical Monitor.

References

[1]Farhet, 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 12, 1998, pp 95–114.

[2]Samareh, J.A., "Discrete Data Transfer Technique for Fluid-Structure Interactions", AIAA-2007-4309, June 2007.

[3]Gray, J., Moore, K.T., and Naylor, B.A., "OpenMAO: An Open Source Framework for Multidisciplinary Analysis and Optimization", AIAA-2010-9101, September 2010.

[4]Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., *Numerical Recipes in Fortran 77*, Cambridge University Press, 1992. pp 413–418.

[5]Hascoe, L. and Pascual, V., "The Tapoenade Automatic Differentiation Tool: Principles, Model, and Specification", *ACM Transactions on Mathematical Software*, Vol 38, No 3, `http://dx.doi.org/10.1145/2450153.2450158`, 2003.

[6]Poirier, D., Allmaras, S.R. , McCarthy, D.R., Smith, M.F., and Enomoto, F. Y., "The CGNS system", AIAA-1998-3007, 1998.