AIAA JOURNAL
Vol. 35, No. 4, April 1997

Conservative Load Projection and Tracking for Fluid-Structure Problems

Juan Raúl Cebral* and Rainald Löhner†
George Mason University, Fairfax, Virginia 22030-4444

The loose coupling of computational fluid dynamics and computational structural dynamics solvers introduces some problems related to the information transfer between the codes. Some techniques developed to solve the problems of the load transfer and interface surface tracking are presented. The main criterion is to achieve conservation of total loads and total energy. The load projection scheme is based on Gaussian integration and fast interpolation algorithms for unstructured grids. The surface tracking algorithm, also based on interpolation, is important for many applications, including aeroelastic deformation of wings due to aerodynamic loads. The methodologies not only improve present fluid–structure interaction simulations, but also increase the range of their applicability. These techniques are of general character and can be used in other multidisciplinary applications as well.

Introduction

THE simulation of fluid–structure interaction problems is becoming of great importance for engineering purposes because both computational fluid dynamics (CFD) and computational structural dynamics (CSD) have reached a very high degree of development.

There are basically two possible types of coupling: strong[1] and loose.[2–5] In the first case, both the fluid and the structure are treated as a single coupled system of equations and solved simultaneously, whereas in the second case, the solution of the fluid and the structure are performed by independent codes coupled together.

The major drawback of the strong coupling is that the matrix for the full system may be ill-conditioned because of the difference in stiffness of the fluid and the solid, and their respective discretizations. Also, it requires major rewriting of the structural and fluid computer programs.

On the other hand, a loose coupling allows the reuse of large fluid and structural solvers, introducing only minor modifications.[2,4–9] Furthermore, the best-suited solution strategy can be selected for each code independently. However, this approach presents some new problems that need to be addressed. One is the problem of the information transfer between the codes. This can be subdivided into load transfer from the fluid to the structure, different time steps, and interface surface tracking. All of them appear as a result of the possibility of having different discretizations for the fluid and solid domains, and their surface interface.

In the loose coupling approach, the CFD and CSD codes are converted into subroutines that are called alternatively from a master program. This master program is in charge of driving the multidisciplinary application as well as transferring the data (boundary conditions) on the CFD/CSD surface interface between the codes. This kind of paradigm has not only the advantage that very specialized codes can be used to solve for the fluid and the structure, without having to rewrite them, but also the problems mentioned earlier can be studied independently, confining them only to the information transfer routines, in the master program.

The purpose of this paper is to present some techniques to deal with the information transfer between fluid and structural codes, and the problems already mentioned. The main task is to introduce algorithms and techniques for the load projection and surface tracking that conserve the total load as well as the total energy. In previous work, this problem 1) did not appear (e.g., same surface discretization for the fluid and the structural domains[2,6–9]), 2) was ignored (with the assumption that the surface elements for the fluid and structure were of similar size[3,8,10]), or 3) was treated by introducing an additional virtual surface between the fluid and structural surfaces.[4,11] As one can see, none of these approaches is general. This paper presents a conservative, monotonic, adaptive Gaussian quadrature to solve the load transfer problem in a general way.

Received Feb. 2, 1996; revision received Oct. 7, 1996; accepted for publication Dec. 26, 1996.

*Research Scientist, Institute for Computational Sciences and Informatics. Member AIAA.
†Professor, Institute for Computational Sciences and Informatics. Member AIAA.

Load Projection

In most cases, the discretization of the fluid domain and the solid domain are quite different. This results in different surface discretizations of the fluid–structure interface. The traditional approach to the solution of this problem has been the interpolation of the fluid loads to the structure surface.[3,8,10] The problem of interpolation between two structured grids is trivial. For unstructured grids, fast and efficient algorithms are also available.[12] The basic idea is, for a given point of a grid, to identify the host element in the other grid, and then interpolate from its nodes to the point (see Fig. 1). Although this approach may work well in many situations, particularly those where grid sizes are comparable, its major drawback is that it is not conservative; thus under certain conditions it may fail. Figure 2 shows an example in one dimension, where information is being lost during the interpolation. For this reason, Guruswamy and Byun[4] introduced a virtual surface as a way to achieve conservative load transfer. The generation of yet another surface is deemed to be an unnecessary complication, particularly for complex geometries.

Conservative Scheme

In this section we propose another approach, which is conservative in the sense that the total load computed on the fluid surface is exactly the same as the total load computed on the solid surface. This technique also leads us to a much better energy conservation across the fluid–structure interface.

To illustrate the basic idea of this method, let us consider only pressures being transmitted from the fluid to the solid, as in the inviscid case, without loss of generality. Suppose p_f denotes the fluid pressure at the interface and p_s the pressure seen by the structure, we desire to have

$$p_s(x) \approx p_f(x) \tag{1}$$

This equality can be approximately satisfied by using a weighted residual method.[13] Multiplying both sides by a set of weighting functions $\{W^i\}$ and integrating over the surface interface Γ yields

$$\int_\Gamma W^i p_s \, d\Gamma = \int_\Gamma W^i p_f \, d\Gamma \tag{2}$$

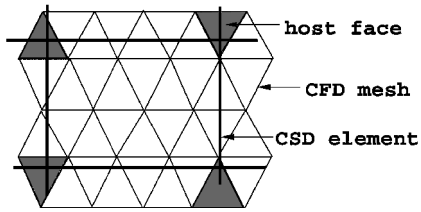


Fig. 1 CFD host elements for four nodes of a CSD quad element.

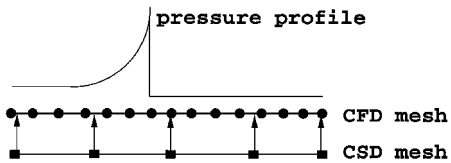


Fig. 2 Case where pressure is not fully transmitted from the fluid to the structure because of the nonconservation properties of the interpolation.

The pressures are approximated using standard shape functions as follows:

$$p_s(x) \approx N_s^j(x)\hat{p}_{sj}, \qquad p_f(x) \approx N_f^j(x)\hat{p}_{fj} \tag{3}$$

In this way, CFD codes based on finite difference, finite volume, finite element, or spectral element approximations can all be accommodated. Taking a Galerkin method $W^i = N_s^i$ in Eq. (2) and inserting the finite element approximations for the pressures, Eq. (2) becomes

$$\int_\Gamma N_s^i N_s^j \hat{p}_{sj}\,\mathrm{d}\Gamma = \int_\Gamma N_s^i N_f^j \hat{p}_{fj}\,\mathrm{d}\Gamma \tag{4}$$

The integral on the left-hand side yields the consistent mass matrix for the solid surface elements M_{cs} and can be computed easily. To solve Eq. (4) for the solid pressures $\hat{p}_{sj}$, the mass matrix has to be inverted:

$$\hat{p}_{sk} = (M_{cs}^{-1})^{ki} r_i \tag{5}$$

where we have defined

$$(M_{cs})^{ki} = \int_\Gamma N_s^i N_s^k\,\mathrm{d}\Gamma \tag{6}$$

and

$$r_i = \int_\Gamma N_s^i N_f^j \hat{p}_{fj}\,\mathrm{d}\Gamma \tag{7}$$

We can prove that this algorithm is conservative in the sense that

$$\int_\Gamma p_f\,\mathrm{d}\Gamma = \int_\Gamma p_s\,\mathrm{d}\Gamma \tag{8}$$

By using the summation property of the shape functions $\sum N_s^i = 1$,

$$\begin{aligned}\int_\Gamma p_s\,\mathrm{d}\Gamma &= \int_\Gamma N_s^j \hat{p}_{sj}\,\mathrm{d}\Gamma = \int_\Gamma \sum N_s^i N_s^j \hat{p}_{sj}\,\mathrm{d}\Gamma \\ &= \sum \int_\Gamma N_s^i N_s^j \hat{p}_{sj}\,\mathrm{d}\Gamma = \sum \int_\Gamma N_s^i N_f^j \hat{p}_{fj}\,\mathrm{d}\Gamma \\ &= \int_\Gamma \sum N_s^i N_f^j \hat{p}_{fj}\,\mathrm{d}\Gamma = \int_\Gamma N_f^j \hat{p}_{fj}\,\mathrm{d}\Gamma = \int_\Gamma p_f\,\mathrm{d}\Gamma\end{aligned} \tag{9}$$

The main question is how to evaluate the integral of Eq. (7). Because the grids are not nested, this is a formidable task. Take, for example, the situation illustrated in Fig. 3, where we have a fine triangular mesh and a quad element of a coarser mesh. This would require the evaluation of the integral on the triangular mesh, but in the region delimited by the quad. The fastest and most accurate way we have found is to use an efficient numerical Gaussian quadrature.[14,15] To perform this integral, we have two possibilities:

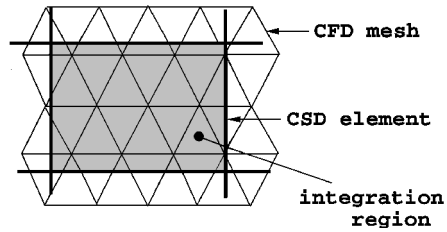
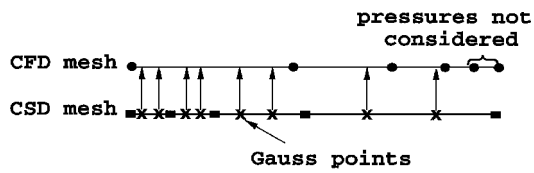
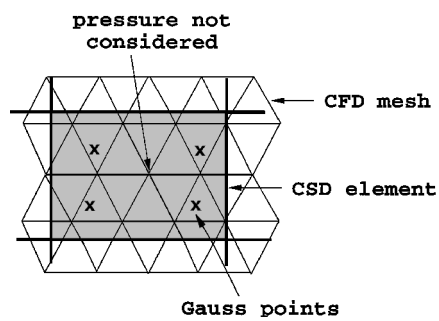


Fig. 3 Integration region for unstructured grids that are not nested.



One-dimensional case



Two-dimensional case

Fig. 4 Introducing Gauss points on the CSD faces is not conservative.

1) Do a loop over the solid faces and introduce a number of quadrature points in each face. In this case, the integral is approximated by

$$r_i = \sum_{\text{solid faces}} \sum_{g}^{n\text{ points}} N_s^i(x_g)\,AW(g)\,\hat{p}_f(x_g) \tag{10}$$

The solid shape functions at the quadrature points g are known, but the fluid pressure has to be interpolated to these points. Although this procedure has been used before,[10,16] it is not conservative because there can be some fluid faces without any quadrature point. An extreme case is shown in Fig. 4, where two Gauss points are introduced in each solid element. Here, the element sizes are very different in each surface discretization, causing some fluid pressures not to be considered. The equivalent two-dimensional case is also shown. By introducing more Gauss points, conservation losses may be reduced, but there is never a guarantee that conservation is preserved accurately enough.

2) Do a loop over the fluid faces and introduce the quadrature points in those faces. In this case, the integral is computed as

$$r_i = \sum_{\text{fluid faces}} \sum_{g}^{n\text{ points}} N_s^i(x_g)\,AW(g)\,\hat{p}_f(x_g) \tag{11}$$

The first sum is over all fluid surface elements, and the second sum is over the quadrature points of the current element; A is the area of the current element, $W(g)$ is the weight of the quadrature point, $\hat{p}_f(x_g)$ is the fluid pressure evaluated at the position of the quadrature point g, and $N_s^i(x_g)$ is the shape function of the solid face that contains the quadrature point g, evaluated at the position of the quadrature point x_g. The pressure at the quadrature points is computed directly from the pressures at the nodes of the fluid element:

$$\hat{p}_f(x_g) = N_f^i(x_g)\hat{p}_{fi} \tag{12}$$

To evaluate the solid shape functions at the quadrature points, it is necessary to know the solid host element of each quadrature

point. This can be done by using optimized searching algorithms.[12] This procedure is conservative because the loop is over the fluid faces, making sure that all pressures are transmitted to the solid. However, some solid points may receive no pressure. Figure 5 shows an example of meshes of widely disparate sizes. This problem can be avoided if enough quadrature points are taken. Therefore, the next question that has to be addressed is how many points need to be introduced in each of the fluid surface elements to achieve an acceptable accuracy. Suppose that we have decided that in a certain face more points are needed; then, we have two possibilities:

1) Use a higher-order Gaussian integration by placing more points at the specified locations and with the corresponding weights, or

2) recursively subdivide the face into smaller elements and introduce the same number of points as before in each subdivision.

The first alternative corresponds to a p-refinement and the second to an h-refinement of the fluid face.[14] The latter case is better because the area of the element is more evenly covered with points and all the points have the same weight. Therefore, this is the method of our choice. The situation is illustrated in Fig. 6, where the problem present in Fig. 5 is overcome by refining an element. Although the equations derived here consider only pressures, they also can be applied to the force vectors with no modifications, to conserve total forces in a viscous case.

Monotonicity-Preserving Scheme

As mentioned before, the solid mass matrix of Eq. (5) has to be inverted to solve for the solid pressures. The first alternative is to lump this matrix, making the inversion trivial. This will give us a low-order solution p^l (dropping the s subindex):

$$M_l \; . p^l = r \tag{13}$$

The second alternative is to use the consistent mass matrix; in this case we will obtain a high-order solution p^h:

$$M_c \; . p^h = r \tag{14}$$

This solution can be computed easily by solving iteratively. If (j) denotes the jth iteration, the pressures can be obtained from

$$(M_l)^{kk} \hat{p}_k^{(j+1)} = r_k + (M_l - M_c)^{ki} \hat{p}_i^{(j)}, \qquad \hat{p}_k^{(0)} = (M_l^{-1})^{kk} r_k \tag{15}$$

Usually, only five iterations are enough for an accurate result. This iterative procedure is readily vectorized and parallelized. Given that, in general, M changes in time, it is clear that this procedure is far superior to any other solution techniques (e.g., L-U decomposition). Although p^h, p^l will yield exact conservation of loads, the high-order scheme may lead to under- and overshoots. This is similar in spirit to CFD techniques for advection-dominated flows, where high-order schemes, although more accurate, produce nonphysical over- or undershoots. In some cases, this loss of monotonicity may be tolerable. However, under certain circumstances (e.g., walls breaking at a certain pressure load), such overshoots would lead to wrong results. For this reason, we try to avoid nonphysical pressures as much as possible and introduce a monotonicity-preserving projection scheme along the lines of flux-corrected transport (FCT),[17] which we call pFCT.

Suppose we have a low-order and a high-order approximation to the projected pressures, computed as explained before [Eqs. (13) and (14)]. The high-order solution may be recovered from

$$M_l \; . p^h = r + (M_l - M_c) \; . p^h \tag{16}$$

This leads to the definition of an antidiffusive flux of the form

$$d = C_{\mathrm{el}} (M_l - M_c) \; . p^h \tag{17}$$

and the FCT solution for the pressure is

$$M_l \; . p^{\mathrm{fct}} = M_l \; . p^l + d \tag{18}$$

The nonlinear limitor factors C_{el} for each element are computed using the same procedures as FEM-FCT.[18]

Summary of Load Transfer Algorithm

The approach described earlier is based on an adaptive refinement of the fluid faces to match the sizes of the solid faces. The measure of the size of an element is taken as the square root of its area. The number of subdivisions needed for a given fluid element is computed by comparing its relative size to the sizes of the solid hosts of its nodes. Also, a minimum number of subdivisions in each face can be imposed. We place either three or four points in each subdivision. The fluid faces are not actually subdivided, only more points are introduced in those whose size is large compared to the corresponding solid faces. In this way, we expect to have a nonzero pressure in all solid points. However, as a safety precaution, we check if any solid point has no pressure. If such a point is found, we increase the minimum number of subdivisions of its fluid host face (and its neighbors if desired), and go back to recompute the Gauss points.

The basic algorithmic steps to be followed can be summarized as follows.

1) For each CFD element:
   Initialize with a minimum number of subdivisions desired.
2) For each CFD element:
   Compute number of subdivisions needed (comparing CFD/CSD sizes).
   Introduce three quadrature points in each subdivision.

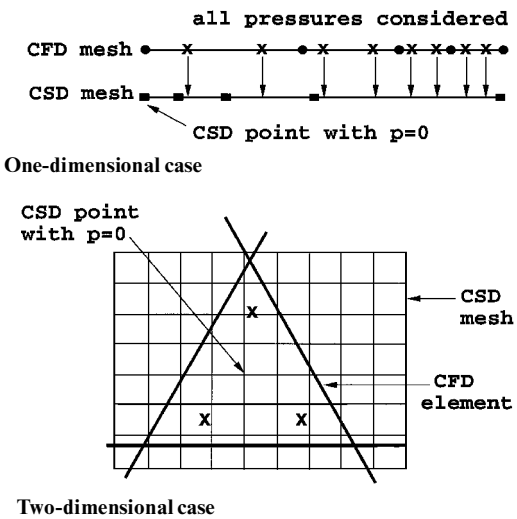


Fig. 5 Introducing Gauss points on the CFD faces is conservative, but some CSD points may receive no pressure.

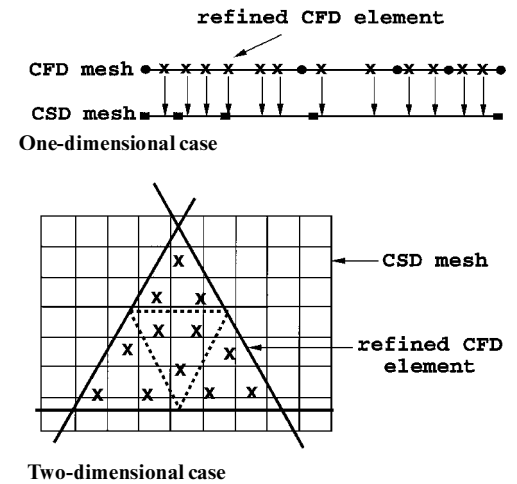


Fig. 6 Introducing Gauss points on the CFD faces that are recursively refined is conservative, and every CSD point receives some pressure.

3) For each quadrature point:
   Find CSD host element (search algorithm).
   Compute CSD shape functions at quadrature point.
4) For each quadrature point:
   Get the CFD pressure.
5) For each CSD point:
   Compute the right-hand side for the CSD pressures.
6) Check if any CSD point has pressure below a specified threshold.
   If so, increment the number of subdivisions of its CFD host face (and its neighbors if desired).
   Go back to 2
7) For each CSD point:
   Get the low-order solution (lumped mass matrix).
   Get the high-order solution (iterate to solve for the consistent mass matrix).
   Limit and solve for pFCT pressures.

Note that the quadrature points need to be recomputed only if the surface grids at the interface change (interconnectivity change, *h*-refinement, remeshing, etc.). Thus, if one stores the Gauss points, the algorithm becomes very efficient (only steps 4–7 have to be executed every time). Also, a renumbering of the Gauss points to minimize cache misses improves the performance on RISC-processor-based machines.[19] For typical production runs,[5,20] the time spent in load transfer, as well as surface tracking (see next section), is less than 1% of the total run time. The surface-grid and Gauss-point storage requirements are negligible when compared with those of the three-dimensional volume grids used for the flow solver, particularly for meshes in excess of 1 Mtet.

Surface Tracking

Another important problem introduced by the loose coupling of CSD and CFD codes is how to make the fluid grid follow the motion and deformations of the solid grid. Here, the easiest way to proceed is to make the fluid mesh follow exactly the solid mesh by interpolating the coordinates of the fluid points to the solid surface.[16] In some cases this technique works very well, e.g., strong shock–object interaction with large plastic deformations. But in other situations, it is not appropriate. For example, in the problem of aeroelastic deformations of a wing,[3,4,7–9] the fluid grid will be three dimensional and fine on the surface. On the other hand, for the structure, the wing can be discretized by taking only a series of planar plates. In this case, it is clear that if we force the fluid mesh to follow the solid mesh exactly, it will result in a wing with no thickness, unacceptable as a model for the fluid. Therefore, other strategies have to be devised.

An alternative to the exact surface tracking described above is to compute the initial vector difference of the fluid and solid grids. Then, as the solid surface moves and deforms, these vectors are rotated and translated accordingly. In this approach, the difference vectors are rotated in exactly the same way as the normals to the solid surface, as illustrated in Fig. 7. The velocities of the solid surface have to be transmitted to the fluid to apply the boundary conditions. If the two surfaces are kept glued, the velocities can be interpolated, but if the surfaces are kept apart, the velocities have to be updated accounting for the rotation of the initial difference vectors. Figure 8 shows a solid under rigid rotation and the transmitted velocities on the fluid mesh, with and without addition of the rotational velocity. The rotational velocity can be obtained from the local rotational velocity of the solid: $\omega = \frac{1}{2}\nabla \times v_s$.

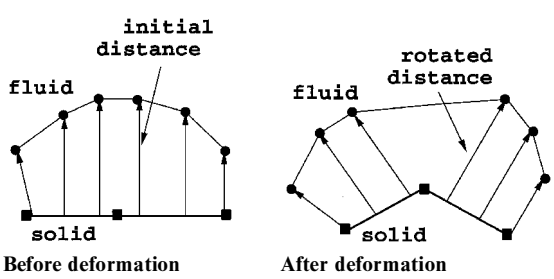


Fig. 7 Surface tracking with initial distance vectors.

Table 1 Pressure distribution on CSD points

Case	Point	Lumped	Consistent	pFCT
a	1	1.0000	0.9861	1.0000
	2	1.1250	1.0070	1.1250
	3	1.5000	1.9856	1.5000
	4	1.1250	1.0070	1.1250
b	1	0.9375	0.2887	0.9375
	2	1.3750	1.9648	1.3750
	3	1.1875	1.2464	1.1875
	4	1.3750	1.9648	1.3750

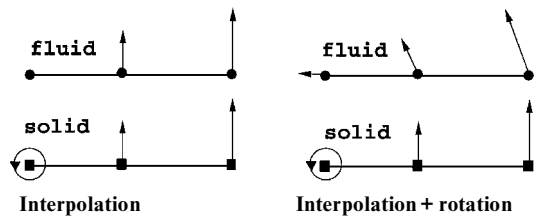


Fig. 8 Addition of the rotational velocities for the surface tracking with initial distance vectors.

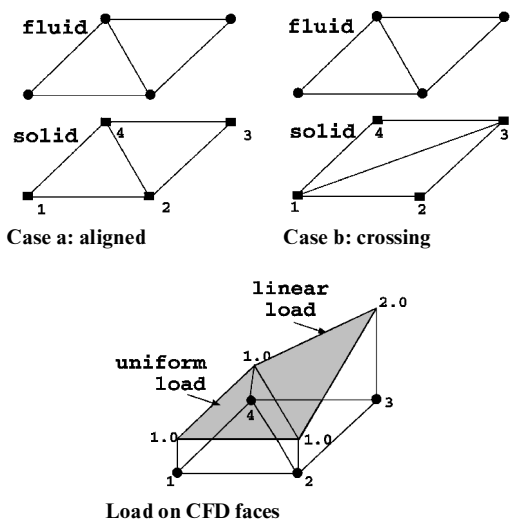


Fig. 9 Two-element fluid–structure interface and loading.

Test Case: Two-Element Interface

In this section, we present a test case to illustrate the differences between the projection schemes, as well as the need for a monotonicity-preserving algorithm. We consider a fluid–structure interface composed of two fluid elements and two solid elements, connected in two distinct ways: aligned (case a) and crossing (case b), as shown in Fig. 9. The fluid pressure distribution is uniform on the first element and linear on the second. The pressures on the CSD points for the two cases, using the lumped mass matrix (low order), the consistent mass matrix (high order), and the pFCT are presented in Table 1. Four quadrature points are introduced in each fluid element.

The element sides are taken of size 1.0 and the pressure at the CFD points is 1.0 and 2.0, as shown in Fig. 9. The total load computed on the CFD surface is 1.1666. For all cases considered in Table 1, the difference between the CFD and the CSD total loads varies from 2.22×10^{-16} to 6.66×10^{-16}. Thus, in all the cases we achieve exact load conservation (to machine roundoff). However, the load distribution and, in particular, the maximum and minimum pressures obtained on the CSD surface are not exactly the same as those on the CFD surface. Furthermore, if the consistent mass matrix is used instead of the lumped mass matrix, we get, as expected, some over- or undershoots with respect to the distribution obtained with the lumped mass matrix. This justifies the introduction of a monotonicity-preserving scheme as the pFCT described earlier. In this particular example, after limiting the high-order scheme, we obtain the same results as in the low-order case.

Numerical Example: Shock–Plate Interaction

This test case consists of a compressible fluid flow in a box, where an elastic plate has been placed on the floor of the box. The geometry definition for the fluid as well as the grid for the structure (plate) are shown in Fig. 10a. The grid for the plate is a structured mesh of 36 quads, and much coarser than the fluid mesh, composed of approximately 250,000 tetrahedra. The triangular surface mesh for the fluid is shown in Fig. 10b. Note the difference in the element size between the two grids. The plate is a square of 1.0×1.0 m and thickness $h = 0.01$ m. Its material properties are taken from mild steel, which in mks units are $E = 200.0\,\mathrm{GN/m^2}$ (Young's modulus), $\nu = 0.31$ (Poisson's ratio), and $\rho = 7840.0\,\mathrm{kg/m^3}$ (density). The fluid is initialized with a strong explosion right before the elastic plate. The density, velocity, and pressure profiles correspond to Sedov's solution,[21] as displayed in Fig. 11a. The notation is the same as used by Landau and Lifshitz.[21] The jump in pressure (p_2/p_{ambient}) is 96.735, the jump in density ($\rho_2/\rho_{\text{ambient}}$) is 5.84, and the fluid velocity at the peak (v_2) is 2602.6 m/s. The fluid is updated using FEFLO97, a linear finite element, edge-based ALE FEM-FCT solver that operates on tetrahedral grids.[20] The structure is modeled using analytical eigenmodes for the plate.[22] A typical run takes less than 5 min on a Cray C90.

Because of the difference in the interface surface discretizations and the sharpness of the pressure profile, very poor load conversion is expected for the interpolation scheme. However, bear in mind that for fluid–structure problems, such size discrepancies are quite common because the analyst has to compromise between model fidelity, computer resources, grid generation requirements, and manhours.

The fluid pressure contours after 80 time steps is shown in Fig. 11b, together with the CSD grid. The pressure contours as seen by the CSD mesh using the interpolation and the pFCT algorithms are shown in Figs. 12a and 12b, respectively. As can be seen, at this particular instant, when the peak of the fluid pressure is not on the CSD grid points, the CSD pressure distribution for the two schemes is quite different. A second comparison between the pFCT projection scheme and the direct interpolation of pressures is shown in Fig. 13, where the total impulse on the plate is plotted as a function of time, for a) the pFCT scheme, b) the direct interpolation, and c) their difference. At the beginning of the simulation, the shock is just entering the plate; at the end, it has just left it.

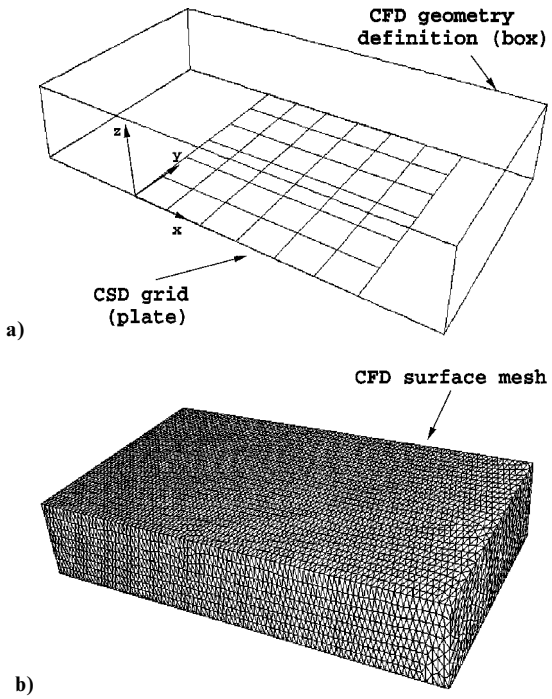


Fig. 10 Problem definition and CSD/CFD grids.

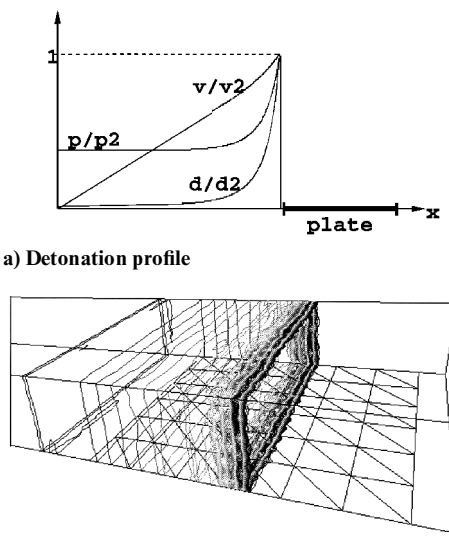


Fig. 11 Density, velocity, and pressure profiles.

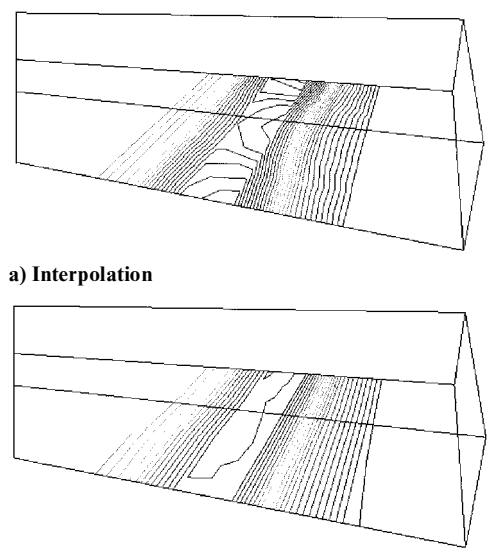


Fig. 12 Solid pressure contours.

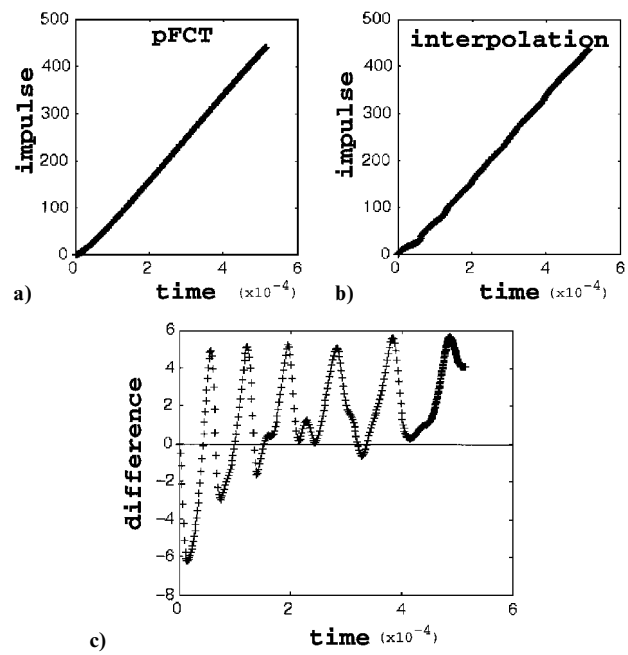


Fig. 13 Comparison of total impulses for pFCT and interpolation.

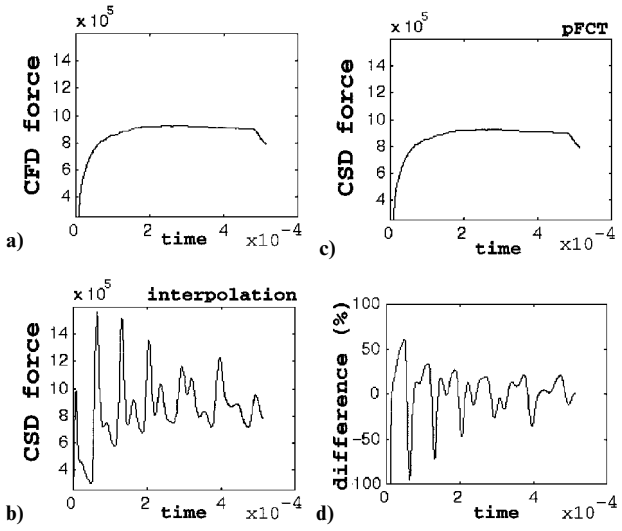


Fig. 14 Comparison of the total forces obtained with pFCT and interpolation.

The total impulse $I(t)$ is computed from the total force F as

$$I(t) = \int_0^t F(t\prime)\,\mathrm{d}t\prime, \qquad F(t) = \int_\Gamma p_s(t)\,\mathrm{d}\Gamma \tag{19}$$

and p_s is obtained from the two mentioned projection schemes. It can be seen that the impulse for the interpolation case is fluctuating, which is to be expected as the shock crosses the structural grid points. The integral of the impulse difference between the pFCT and the interpolation scheme is increasing in time, indicating a loss of impulse for the latter. Note also that the difference is larger when the shock is passing over the CSD grid points, giving a much larger pressure than when it is in between. These differences in the load transmitted to the structure will affect its behavior. If interpolation is used, extreme pressures appear, which may cause fake plasticity regions and fake accelerations over short periods of time. In a third comparison, we consider the total force on the plate as a function of time. Figure 14a shows the total force computed at the CFD–CSD interface, but on the CFD mesh. Figures 14b and 14c show the total force at the interface on the CSD mesh as computed by the interpolation and the pFCT, respectively. The difference between the total force on the CFD mesh and the total force on the CSD mesh computed with the pFCT is always zero, implying conservation to machine roundoff. The relative difference with the interpolation scheme is shown in Fig. 14d, as a percentage of the CFD force. It is clear that in this case, the load is not conserved, giving very large differences (60%), especially when the shock wave is on top of the CSD grid points.

Conclusions

Fluid–structure interaction problems are best tackled by a loose coupling of previously developed fluid and structural codes. However, this approach introduces some new problems to be solved, namely load projection and surface tracking.

A conservative technique for the load transfer, based on adaptive Gaussian integration and interpolation on unstructured grids, is presented. The surface tracking problem is very important for many applications, especially in aeroelasticity, where the fluid grid and the solid grids have to be separated. The two problems converge at this point because the conservation of total loads and a good surface tracking algorithm must successfully account for the energy conservation.

The transmission of pressures in one direction (CFD $\longrightarrow$ CSD) and of velocities in the opposite direction (CSD $\longrightarrow$ CFD) is consistent with the fact that pressure and velocity are conjugate variables, thus allowing conservation of energy.

With the methodologies presented, the range of applications that can be simulated accurately with loosely coupled fluid–structure codes has been increased considerably. In particular, cases in which large size differences between the surface elements of the CFD and CSD discretizations are present can now be treated accurately in a straightforward way. This is achieved without recourse to an artificial, or virtual, third surface grid to connect the CFD and CSD domains.

Future work will center on coupling of implicit CFD and CSD codes for aeroelastic cases and porting of the coupling methodology to parallel machine environments.

Acknowledgments

This research was funded by the U.S. Air Force Office of Scientific Research under Contract F-49620-94-1-0119. Leonidas Sakell was the Technical Monitor.

References

[1]Felker, F. F., "Direct Solution of Two-Dimensional Navier–Stokes Equations for Static Aeroelasticity Problems," *AIAA Journal*, Vol. 31, No. 1, 1993, pp. 148–153.

[2]Guruswamy, G. P., "ENSAERO—A Multidisciplinary Program for Fluid/Structural Interaction Studies of Aerospace Vehicles," *Computing Systems in Engineering*, Vol. 1, Nos. 2–4, 1990, pp. 237–256.

[3]Guruswamy, G. P., "Unsteady Aerodynamic and Aeroelastic Calculations for Wings Using Euler Equations," *AIAA Journal*, Vol. 28, No. 3, 1990, pp. 461–469.

[4]Guruswamy, G. P., and Byun, C., "Fluid-Structural Interaction Using Navier–Stokes Flow Equation Coupled with Shell Finite Element Structures," AIAA Paper 93-3087, July 1993.

[5]Löhner, R., Yang, C., Cebral, J. R., Baum, J. D., Luo, H., Pelessone, D., and Charman, C., "Fluid-Structure Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids," *Computational Fluid Dynamics Review 1995*, Wiley, New York, 1995, pp. 755–776.

[6]Farhat, C., and Lin, T. Y., "Transient Aeroelastic Computations Using Multiple Moving Frames of Reference," AIAA Paper 90-3053, 1990.

[7]Batina, J. T., Bennet, R. M., Seidel, D. A., Cunningham, H. J., and Bland, S. R., "Recent Advances in Transonic Computational Aeroelasticity," *Computers and Structures*, Vol. 30, Nos. 1 and 2, 1988, pp. 29–37.

[8]Rausch, R. D., Batina, J. T., and Yang, H. T. Y., "Three-Dimensional Time-Marching Aeroelastic Analyses Using an Unstructured-Grid Euler Method," *AIAA Journal*, Vol. 31, No. 9, 1993, pp. 1626–1633.

[9]Farhat, C., and Lin, T. Y., "Structure-Attached Corotational Fluid Grid for Transient Aeroelastic Computations," *AIAA Journal*, Vol. 31, No. 3, 1993, pp. 597–599.

[10]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.

[11]Appa, K., "Finite Surface Spline," *Journal of Aircraft*, Vol. 26, No. 5, 1989, pp. 495, 496.

[12]Löhner, R., "Robust, Vectorized Search Algorithms for Interpolation on Unstructured Grids," *Journal of Computational Physics*, Vol. 118, 1995, pp. 380–387.

[13]Zienkiewicz, O. C., and Morgan, K., *Finite Elements and Approximations*, Wiley, New York, 1983, pp. 38–95.

[14]Zienkiewicz, O. C., and Taylor, R. L., *The Finite Element Method*, 4th ed., Vols. 1 and 2, McGraw–Hill, New York, 1989, Chap. 8.

[15]Bathe, K. J., *Finite Element Procedures in Engineering Analysis*, Prentice–Hall, Englewood Cliffs, 1982, pp. 268–294.

[16]Löhner, R., Yang, C., Cebral, J. R., Baum, J. D., Luo, H., Pelessone, D., and Charman, C., "A Loose Coupling Algorithm for Fluid-Structure Interaction Simulations," *Proceedings of the Eighth Annual Idaho National Engineering Laboratory Computing Symposium* (Idaho Falls, ID), 1994, pp. 10.3–10.14.

[17]Zalesak, S. T., "Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids," *Journal of Computational Physics*, Vol. 31, 1979, pp. 335–362.

[18]Löhner, R., Morgan, K., Peraire, J., and Vahdati, M., "Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier–Stokes Equations," *International Journal for Numerical Methods in Fluids*, Vol. 7, 1987, pp. 1093–1109.

[19]Löhner, R., "Some Useful Renumbering Strategies for Unstructured Grids," *International Journal for Numerical Methods in Engineering*, Vol. 36, 1993, pp. 3259–3270.

[20]Baum, J. D., Luo, H., Löhner, R., Yang, C., Pelessone, D., and Charman, C., "A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck," AIAA Paper 96-0795, Jan. 1996.

[21]Landau, L. D., and Lifshitz, E. M., "Fluid Mechanics," *Course of Theoretical Physics*, 2nd ed., Vol. 6, Pergamon, New York, 1987, Chap. 10.

[22]Timoshenko, S. P., and Goodier, J. N., *Theory of Elasticity*, 3rd ed., McGraw–Hill, New York, 1970, Chap. 3.

R. K. Kapania
Associate Editor