FLECS, a Flexible Coupling Shell Parallel Application to Fluid-Structure Interaction

Margreet Nool*, Erik Jan Lingen†, Sander van Zuijlen**, Martijn Stroeven† and Hester Bijl**

*CWI, Dept. Computing and Control, Amsterdam, The Netherlands, email=Margreet.Nool@cwi.nl
†Habanera Software company, Delft, The Netherlands
**TU Delft, Fac. Aerospace Engineering, Delft, The Netherlands

Abstract. In this paper we discuss the second version of FLECS, a generic, open-source coupling shell that can be used to join two or more arbitrary solvers. In general multidisciplinary computations are very computing-intensive. A remedy against long computing times is large-scale parallelism. The challenge of the present parallelization work is to obtain acceptable computing times and to get rid of severe memory requirements that exist on sequential machines, for the generic flow problems at hand. The aim is to provide a flexible platform for developing new data transfer algorithms and coupling schemes.

Keywords: Parallel Computing, Coupling Shell, Fluid-Structure Interaction
PACS: 0.2.60.Cb,0.2.60.Dc,0.2.60.Ed

1. INTRODUCTION

Numerical simulations involving multiple, physically different domains, like a solid structure and a fluid, can be solved effectively by coupling multiple simulation programs, or *solvers*. Each solver deals with one particular physical domain, applying the numerical algorithms that are most efficient for that domain. The solvers regularly exchange data to take into account the effects of the other domains. The coordination of the different solvers is commonly handled by a *coupling shell*. The majority of coupling shells are embedded subprograms that have been developed for coupling two specific solvers. One exception is the coupling library MPCCI (Mesh based Parallel Code Coupling) [3], which can be used as a separate program. Although MPCCI is relatively easy to use and provides many advanced features, it is less suitable for a scientific research community that is aimed at developing new data transfer algorithms. Numerical acceleration algorithms, like Krylov and multilevel methods - urgently required for efficiency - are not incorporated. Moreover, since MPCCI only provides the binary code, the user can not modify the implementation schedule of MPCCI.

The coupling shell FLECS (Flexible Coupling Shell) synchronizes the execution of the solvers and handles the transfer of data from one physical domain to another. In a coupled fluid-structure simulation, for instance, the coupling shell transfers pressures from the fluid to the surface of the structure, and velocities and displacements from the structure surface to the fluid. In addition to accurate coupling in space, it is possible to reduce the partitioning errors in time by using specially designed high-order time integration methods. Moreover, FLECS can be combined with a multi-level acceleration technique, based on a presumed existing multi-grid solver for the flow domain, with the Aitken underrelaxation technique [6].

In contrast to the first version of FLECS [2, 4], the second version can take full advantage of parallel computers, supporting both parallel solvers and parallel transfer algorithms (see Fig. 1). This makes it possible to use FLECS for large-scale 3-D coupled simulations involving transfer algorithms that are accurate in space and time. FLECS uses MPI-2 to achieve good performance while reducing the dependencies between coupled solvers. That is, each solver can run in its own environment and memory space, and use its own start-up procedure. FLECS provides language bindings for C and Fortran 90 so that it can be easily used in a wide range of solvers.

The effectiveness of FLECS will be tested with two parallel transfer algorithms: one based on a Radial Basis Function (RBF) method and one based on a Nearest Neighbor (NN) Method.

2. DESIGN OVERVIEW

FLECS is decomposed into a *client library* that is to be called from the solver programs, and a *coupling server* that coordinates the execution of the solvers, takes care of the coupling of the domains and handles the transfer of data between the solvers.

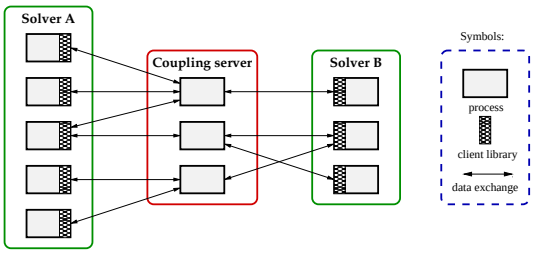

FIGURE 1. Schematic representation of second version of FLECS

2.1. The Client Library

The client library provides subroutines with which a solver can open and close connections with one or more coupling servers; coordinate the execution with those servers; transfer data between physical domains; and perform a number of miscellaneous tasks, including error handling.

Before a solver can transfer data from its own domain to another, it needs to create one or more *item sets* that describe the grid on its domain interface. Each item in an item set represents an entity such as a node, cell or element, and is identified by a unique integer *identifier*, also called its ID. Except for the item IDs, FLECS does not store any data associated with the items in an item set; it simply passes the item data from a solver to a coupling server without interpreting those data.

If a solver program comprises multiple parallel processes, then each process may define its own set of items corresponding with a part of the interface grid. The processes may redistribute the items during the execution of a coupled simulation to achieve a better load balance.

2.2. The Coupling Server

The coupling server (see Fig. 2) consists of two parts: one that handles the initialization of the server and the communication between the server and the client library, and one that transfers data from one physical domain to another. This part of the server is based on a plug-in architecture, that makes it easy to implement new transfer algorithms.

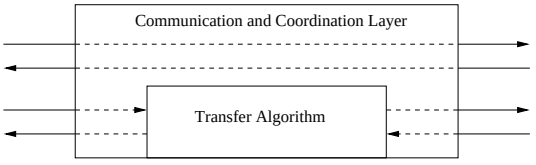

FIGURE 2. Schematic representation of the coupling server

For each item set created by a solver program, a corresponding item set is created by the server. If the solver comprises multiple parallel processes, then FLECS determines an initial distribution of the items over the server processes. This distribution may be changed by a transfer algorithm to achieve a better load balance and/or to reduce the amount of communication between the server and the solver processes.

3. AN APPLICATION TO FLUID-STRUCTURE INTERACTION

De Boer's thesis [2] contains a detailed study of three different techniques to transfer information between non-matching meshes in fluid-structure interaction (FSI) computations: NN interpolation, the weighted residual method and the RBF interpolation. Two transfer techniques are parallelized and will be added to the FLECS package: the RBF method and the NN method.

3.1. Radial Basis Function algorithm

The RBF methods appear to be very suitable because of their high accuracy and efficiency. No orthogonal projection and search algorithms are needed, but the computation involves the inversion of a matrix. The solvers exchange data to take into account the effects on the other domain. The coupling methods can be formulated as

$$\mathrm{d}\mathbf{x}_f = \mathbf{H}_{fs}\mathrm{d}\mathbf{x}_s,\ \mathbf{p}_s = \mathbf{H}_{sf}\mathbf{p}_f, \tag{1}$$

with d$\mathbf{x}$ the vector with the displacements and $\mathbf{p}$ the pressure in the fluid or structure points at the discrete interfaces, and, $\mathbf{H}_{fs} \in \mathbb{R}^{n_f \times n_s}$ and $\mathbf{H}_{sf} \in \mathbb{R}^{n_s \times n_f}$ are transformation matrices. The numbers n_f and n_s, the numbers of flow and structure points on the fluid-structure interface, respectively, are usually very small compared to the total number of structure and flow points.

Equation (2.38) in [2] describes how to calculate the coupling matrix. We first compute the matrix $\hat{H}_{sf}$

$$\hat{H}_{sf} = \begin{bmatrix} \Phi_{fs} & Q_s \end{bmatrix} \cdot \begin{bmatrix} \Phi_{ff} & Q_f \\ Q_f^T & 0 \end{bmatrix}^{-1} = \tilde{\Phi}_{fs} . \tilde{\Phi}_{ff}^{-1}, \tag{2}$$

and then the transformation matrix $\mathbf{H}_{sf}$ is defined as the first n_s rows and n_f columns of the matrix $\hat{H}_{sf}$. The matrix Φ_{fs} contains the evaluation of the basis function $\phi_{f_i,s_j} = \phi(\|\mathbf{x}_{f_i} - \mathbf{x}_{s_j}\|)$. The matrices $\mathbf{x}_f$ and $\mathbf{x}_s$ contain the coordinates of the centers in which the interface values are known. The matrix Q_s is an $n_s \times (d+1)$ matrix with d the dimension of the problem. For more details we refer to [2].

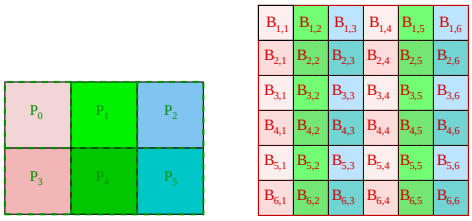

FIGURE 3. (*left*) 6 Processes mapped to a 2x3 process grid and (*right*) 2D-block-cyclic distribution on a 2x3 process grid

We decided to use the software package SCALAPACK, a library of high-performance linear algebra routines, developed for distributed memory systems. The distributed memory version of BLAS, called the PBLAS, is the building block of SCALAPACK routines. Communication between the blocks will be performed by BLACS, a set of Basis Linear Algebra Communication Subprograms. SCALAPACK is portable on any computer or network of computers that supports MPI.

The calculation of the coupling matrix $\mathbf{H}_{sf}$ consists of the following steps: the creation of the (symmetric) matrix $\tilde{\Phi}_{ff}$, the creation of the matrix $\tilde{\Phi}_{fs}$, the computation of the inverse of matrix $\tilde{\Phi}_{ff}$, and the matrix-matrix multiplication to achieve $\hat{H}_{sf} = \tilde{\Phi}_{fs} \times \tilde{\Phi}_{ff}^{-1}$. The first n_s rows and n_f columns form the matrix $\mathbf{H}_{sf}$. The computation of the coupling matrix $\mathbf{H}_{fs}$ goes along similar lines. The SCALAPACK routine PDGESV is used for computing the inverse of the matrix $\tilde{\Phi}_{ff}$ in parallel. The matrix-matrix product can be computed using the PBLAS routine PDGEMM.

3.2. Data distribution

For the dense algorithms implemented in SCALAPACK, the block-cyclic data layout [1] has been selected because of its good scalability, load balance and communication properties, see also Fig. 3.

The p processes of an abstract parallel computer are mostly represented as a 1D linear array of processors. It is often more convenient to map this array into a 2D rectangular grid, or *process* grid. The matrix will be divided into square blocks, e.g., of size 64×64, as recommended in the SCALAPACK User's Guide. Then its blocks will be distributed along the processes according to the colors in the picture. The first block column will be distributed over the processes p_0 and p_3, the next block column over p_1 and p_4 etc.

On each processor $p_i, (0 \leqslant i < p)$, the elements of matrix $\tilde{\Phi}_{ff}$ can be calculated accordingly to the 2D-block cyclic distribution without communication. For the ease of computing, the coordinates $\mathbf{x}_f$ must be present on every processor. Moreover, if the coordinates of $\mathbf{x}_s$ are available too, also the computation of matrix $\tilde{\Phi}_{fs}$ requires no communication.

The distribution of the coupling matrices imposes demands on the coding rules of the input $\mathrm{d}\mathbf{x}_s$ and $\mathbf{p}_f$ and output matrices $\mathrm{d}\mathbf{x}_f$ and $\mathbf{p}_s$ of Eq. (1). It is not expected that these rules agree with the distribution of the pressure and displacement matrices of the parallel solvers. With the help of the SCALAPACK matrix distribution routine PDGEMR2D any block cyclic distributed matrix can be copied to any other block-cyclically distributed matrix, even to a global matrix with block sizes equal to the matrix size. In other words, we are able to map the block-cyclic into a non-cyclic storage on a single process.

3.3. Performance of parallel RBF method

As a test example we choose a flow around a flexible wing (Fig. 4). It is the AGARD 445.6 2.5-foot semi-span model [5], which is often used as a benchmark case for aeroelastic codes. The coupling has been performed with an RBF method with compact support, the so-called CP C^2 with a radius of 0.1. Only one *full step* according to Eq. (1) has been executed: the computations of the matrices $\mathbf{H}_{fs}$ and $\mathbf{H}_{sf}$ and the flow displacements and the structure pressure. The flow mesh has $n_f = 30045$ and the structure mesh has $n_s = 913$ interface points, respectively. This implies that, as part of the calculation of the coupling matrices, the inverses of the matrices $\hat{\Phi}_{ff}$ of size 30050×30050 and $\hat{\Phi}_{ss}$ of 917×917 have to be computed. As a block size for $\hat{\Phi}_{ff}$ we choose 64×64 resulting into 470×470 number of blocks. For $\hat{\Phi}_{ss}$ we reduce the block size to 32×32 to increase the parallelism.

FIGURE 4. Fluid mesh of the AGARD 445.6 2.5-foot semi-span model

It took us a lot of effort to achieve scalable performance. In order to achieve good scalable performance, we experiment with all kinds of configurations of nodes, cores and threads. Finally, the wall clock time for one full step was reduced to less than four minutes on a grid of 12×12 processors. The execution was performed on a Linux cluster with 715 compute nodes. For our application we used 18 nodes with 2*quad-cores, where each node has a memory of 24 Gb. Obviously, the computing time is completely dominated by the computation of the inverse of $\tilde{\Phi}_{ff}$ as can be seen in Table 1. In comparison

TABLE 1. Wall clock times for parts of the calculation of the coupling matrices $\mathbf{H_{sf}}$ and $\mathbf{H_{fs}}$ and the updates of the pressure in the structure interface points and the displacements in the flow interface points

grid	$\tilde{\Phi}_{ff}$	$\tilde{\Phi}_{fs}$	Inv($\tilde{\Phi}_{ff}$)	$\hat{H}_{sf}$	$\mathbf{H}_{sf}\mathbf{p}_f$
4x4	23.27	.55	1022.05	52.60	.48e-1
6x6	8.84	.26	511.64	21.58	.15e-0
8x8	4.78	.13	323.45	12.11	.35e-1
10x10	4.66	.11	262.50	10.51	.26e-0
12x12	3.88	.09	188.08	8.12	.16e-0
grid	$\tilde{\Phi}_{ss}$	$\tilde{\Phi}_{sf}$	Inv($\tilde{\Phi}_{ss}$)	$\hat{H}_{fs}$	$\mathbf{H}_{fs}\mathbf{p}_s$
4x4	.16e-1	.52e-0	10.25	2.45	.55e+1
6x6	.65e-2	.22e-0	12.76	4.09	.41e-0
8x8	.42e-2	.15e-0	8.05	2.48	.15e+1
10x10	.25e-2	.79e-1	8.09	1.47	.65e-0
12x12	.23e-2	.63e-1	6.05	2.04	.92e-0

with the original MATLAB code, the execution of a single step on a desktop computer took several days.

3.4. Nearest Neighbor method

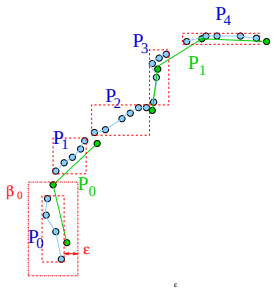

FIGURE 5. Example of Nearest Neighbor implementation using bounding boxes

The NN method is a very simple method to transfer data from mesh **A** to mesh **B**. A search algorithm determines the point x_A in mesh **A** that is closest to a given point x_B in mesh **B**. The variables in x_B are then assigned the same values as in x_A. Note that for the NN method the transformation matrices become Boolean matrices.

In case of parallel solvers at both sides, the points of the meshes **A** and **B** are distributed over different processes, the search process for the closest neighbor becomes more complicated. For each process, a minimal rectangular bounding box is calculated, which contains all points present on that processor. This red dashed box (Fig. 5) will be extended by a strip in all directions. The results of the obtained boxes $\beta_i, 0 \leqslant i < p$, are of importance to determine which processes have to communicate with each other to search for nearest neighbors. The size of this strip ε should be a parameter of the method. If the strip size ε is too small, it may occur that a nearest neighbor will be missed. On the other hand, when it is chosen too large, more communication will take place then strictly necessary.

3.5. Conclusions

Parallel FLECS appears to be a valuable extension of initial FLECS. In the parallel execution no longer bottlenecks will arise. The nearest neighbor method illustrates how communication can be reduced between the solvers and the coupling server. If this search method is not restricted to the interface items, but instead the whole domain is considered, e.g., in case of moving grids, the parallelism in the coupling server is even more favorable. The RBF method shows that for huge problems, i.e., more than 30.000 interface points, memory requirements and acceptable wall clock time urges for parallelism.

ACKNOWLEDGMENTS

Funding for this work was provided by the National Computing Facilities Foundation (NCF), under project numbers NRG-2005.03 and NRG-2007.05. The authors want to acknowledge and thank the High Performance & Networking group of SARA for their help and advice for running the codes on the Dutch National Supercomputer, named *lisa*.

REFERENCES

1. L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, 1997.
2. A. de Boer. *Computational fluid-structure interaction, Spatial coupling, coupling shell and mesh-deformation*, PhD Thesis, TU Delft, 2008.
3. MpCCI, The Fraunhofer-Institute for Algorithms and Scientific Computing (SCAI). *Multidisciplinary Simulation through Code Coupling*, see http://www.scai.fraunhofer.de/mpcci.html
4. Margreet Nool, Erik Jan Lingen, Aukje de Boer and Hester Bijl. *Flecs, a Flexible Coupling Shell Application to Fluid-Structure Interaction*, Lecture Notes in Computer Science, 2007
5. E.J. Yates jr. *AGARD standard aeroelastic configurations for dynamic reponse. candidate configuration I.-Wing 445.6*, AIAA, Tech. Rep. Technical Memorandum 100492, 1987.
6. A.H. van Zuijlen and H. Bijl *Multi-level acceleration for sub-iterations in partitioned fluid-structure interaction*, CP1168, Vol.2, Numerical Analysis and Applied Mathematics, IC 2009.