COMPDYN 2011
III ECCOMAS Thematic Conference on
Computational Methods in Structural Dynamics and Earthquake Engineering
M. Papadrakakis, M. Fragiadakis, V. Plevris (eds.)
Corfu, Greece, 25–28 May 2011

ADVANCED PARALLEL COMPUTING FOR EXPLOSIVE FLUID-STRUCTURE INTERACTION

V. Faucher[1]

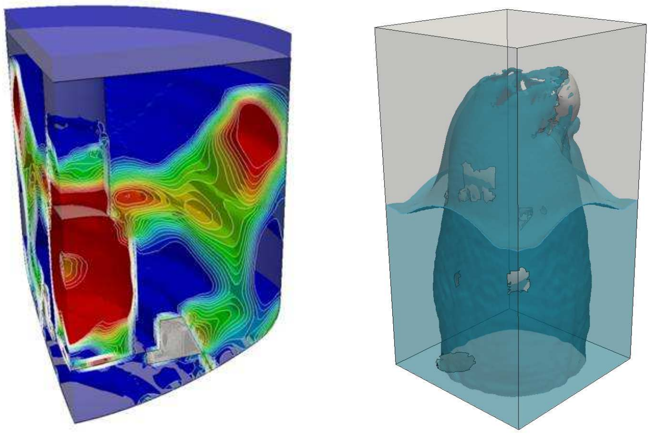
[1] CEA, DEN, DANS, DM2S, SEMT, DYN
CEA/Saclay, 91 191 Gif sur Yvette Cedex, France
e-mail: vincent.faucher@cea.fr

Keywords: Fast transient dynamics, fluide structure interaction, kinematic links.

Abstract. *Present contribution exposes current R&D issues in the framework of parallel computing for fast transient dynamics involving fluid-structure interaction and coupled models. In particular, simulation of accidental explosive situations often requires simultaneous representation of both fluids with multiple components and structures that may undergo large deformation down to complete ruin. Is especially considered the generic treatment of kinematic links by means of Lagrange Multipliers , and the parallel concurrency it implies amoung solving algorithms throughout the simulation. Current research and investigated solutions are illustrated with industrial examples performed with EUROPLEXUS software.*

1 INTRODUCTION

Simulation of fast transient phenomena arising from accidental situations often requires simultaneous modeling of fluids, both liquids and gases, interacting with surrounding structures that may undergo large deformations down to complete ruin. This yields couplings between various kinds of models, such as Finite Elements, Finite Volumes or SPH particles for the fluids, and Finite Elements, SPH particles or Discrete Elements for the structures (cf. figure 1). Examples of such simulations in explosive fluid-structure interaction are given in [1][2].



(a) Explosion in a vessel with internal structures

(b) Explosion et fragmentation d'un bidon immergé dans un liquide

Figure 1: Examples of explosive FSI simulations.

Kinematic links between various models is thus a fundamental issue and the choice is made in EUROPLEXUS to verify the link equations exactly by means of Lagrange Multipliers, avoiding the use of user defined penalty parameters to simplify the input process and prevent any dependency of the solution on non-physical parameters. From the algorithmic point of view, this introduces a strong specificity for the temporal solver: time integration scheme is classically central differences explicit scheme, but solving a linear system at each step is necessary to compute link forces, leading to a partially implicit solving procedure.

Parallel handle of both building and solving the system giving Lagrange Multipliers for all the kinematic connections available in the program competes with classical parallel algorithms used in fast transient dynamics software and is addressed by current paper.

2 GENERAL ALGORITHM

2.1 Dynamic equilibrium for fluid-structure interaction with explicit time integration

Conservation of momentum at time t^{n+1} of the time integration process, all quantities assumed to be known at times t^n (displacements, for structure only) and $t^{n+1/2}$ (velocities), takes general form:

$$\begin{aligned}
&\mathbf{M}^{n+1}\ddot{\mathbf{U}}^{n+1}+\mathbf{F}_i\left(\mathbf{U}^n,\dot{\mathbf{U}}^{n+1/2}\right)+\mathbf{C}^{n+1^T}\mathbf{\Lambda}^{n+1}=\mathbf{F}_{ext}^{n+1}\\
&\mathbf{C}^{n+1}\dot{\mathbf{U}}^{n+3/2}=\mathbf{B}^{n+1}\\
&\dot{\mathbf{U}}^{n+3/2}=\dot{\mathbf{U}}^{n+1/2}+\Delta t\cdot\ddot{\mathbf{U}}^{n+1}
\end{aligned} \tag{1}$$

Forces $\mathbf{F}_i$ represent internal forces for the structure or transport forces for the fluid. The system is completed for the fluid by conservation of mass and energy equations, written in EUROPLEXUS using a Finite Volume formalism, i.e. computing fluxes of quantities through faces of the fluid mesh cells. $\mathbf{M}^{n+1}$ is a diagonal matrix.

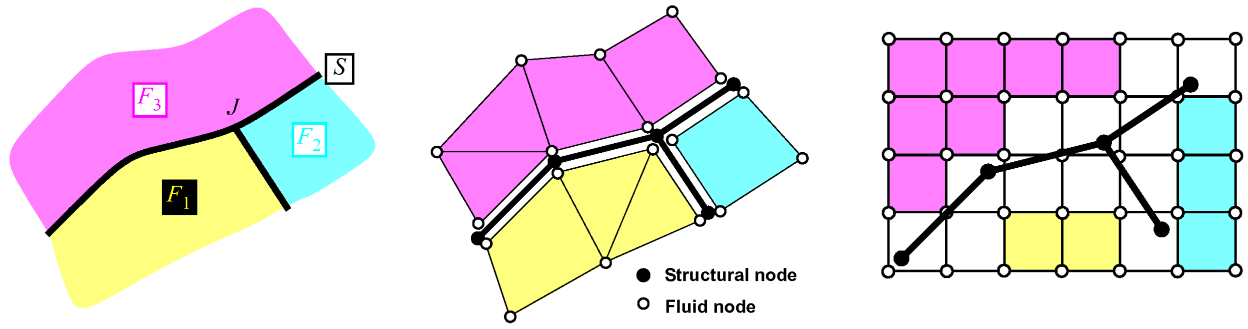
Matrix $\mathbf{C}^{n+1}$ is the kinematic connections matrix, linking velocities of both fluid and structural nodes, whose construction is presented in next paragraph. Its dimension, profile and coefficients are all variable with time.

Before computing nodal accelerations allowing advancing to next time-step in the simulation, Lagrange Multipliers are obtained by a condensation procedure, yielding a system of the form:

$$\mathbf{C}^{n+1}\mathbf{M}^{{n+1}^{-1}}\mathbf{C}^{{n+1}^{T}}\mathbf{\Lambda}^{n+1} = \mathbf{H}^{n+1}\mathbf{\Lambda}^{n+1} = \mathbf{S}^{n+1} \tag{2}$$

2.2 Writing fluid-structure interaction links

Two types of fluid-structure links are considered in classical transient explosive simulations [1][3]. First is conform node-to-node links, when the structure coincides with fluid domain's envelop. Second is diffuse links for immersed structures, for which enforcing a conform mesh between fluid and structure generates large extra work for data sets creation and lack of robustness for ALE rezoning of the fluid grid. Figure 2 illustrates both types of links.



(a) Interface F/S continue (situation de reference)

(b) Interface discrétisée avec maillages conformes

(c) Interface discrétisée avec liaison diffuse

Figure 2: Fluid-structure interaction links.

Interaction with conform meshes produces permanent links, whose coefficients are variable along with direction normal to the structure. Diffuse interaction with topologically uncoupled meshes generates links whose support and coefficients are variable with time.

Second type of link is mandatory to deal with problems where structural fragmentation occurs, to simulate fluid flow through created openings, as shown on figure 1-b or on figure 3.

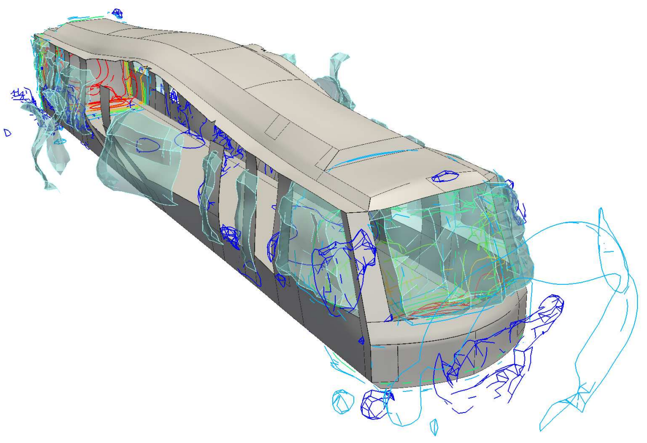


Figure 3: Simulation of an explosion in a metro carriage.

3 PARALLEL SOLUTION WITH DISTRIBUTED MEMORY

3.1 Principle and problematic of *remote* links

Parallel solution in EUROPLEXUS relies on domain decomposition, splitting elements upon available processing units with minimum interface size between subdomains. Formally, system (1) becomes, for example in the case of 3 subdomains:

$$\begin{aligned}
&(a)\quad \begin{bmatrix} \mathbf{M}_1 & & \\ & \mathbf{M}_2 & \\ & & \mathbf{M}_3 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{U}}_1 \\ \ddot{\mathbf{U}}_2 \\ \ddot{\mathbf{U}}_3 \end{bmatrix} + \begin{bmatrix} \mathbf{F}_{i1} \\ \mathbf{F}_{i2} \\ \mathbf{F}_{i3} \end{bmatrix} + \mathbf{C}^T \begin{bmatrix} \mathbf{\Lambda}_1 \\ \mathbf{\Lambda}_2 \\ \mathbf{\Lambda}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{ext1} \\ \mathbf{F}_{ext2} \\ \mathbf{F}_{ext3} \end{bmatrix} \\
&(b)\quad \mathbf{C} \begin{bmatrix} \ddot{\mathbf{U}}_1 \\ \ddot{\mathbf{U}}_2 \\ \ddot{\mathbf{U}}_3 \end{bmatrix} = \mathbf{B}
\end{aligned} \tag{3}$$

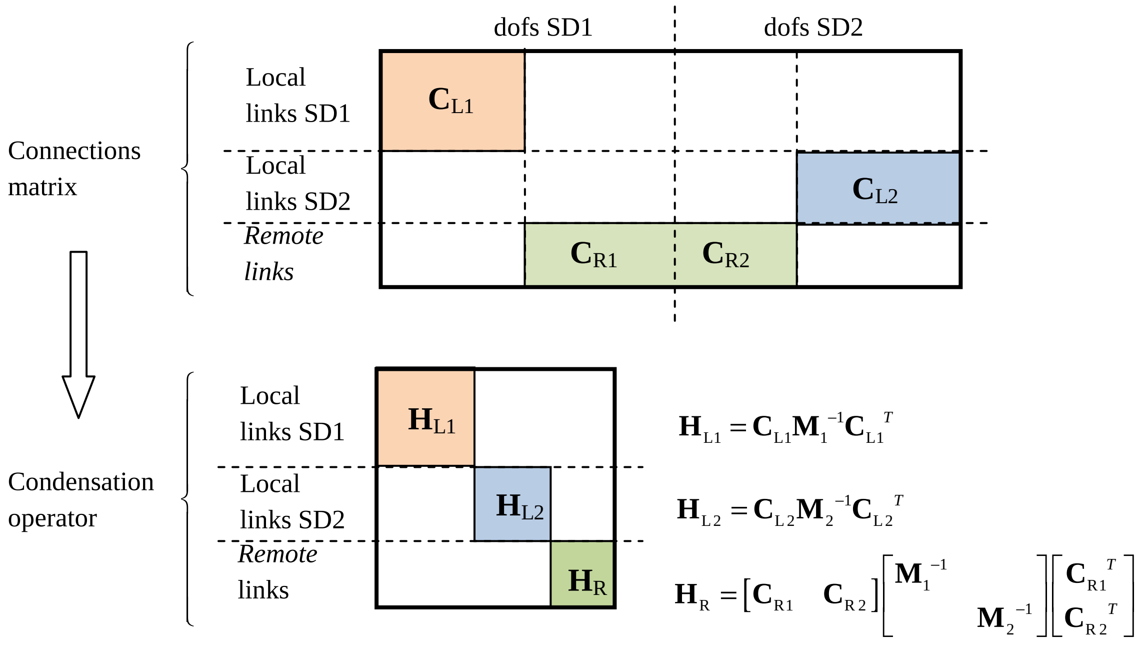
Parallel acceleration is first achieved by distribution of the work for both internal forces vector computation and fluid-structure links building.

However, connections matrix **C** must be considered at global level for the multi-domain system, since links may couple nodes belonging to different subdomains, for example with large displacements of structural fragments through fluid domain. Such links are called *remote* links, to emphasize the fact that they concerns non-local quantities on subdomains.

The problematic associated to these *remote* links is the global treatment they require.

In the rare situation where no link couples degrees of freedom belonging to different subdomains (i. e. no *remote* links in the model), system (3) can be uncoupled and parallelism is straightforward. On the contrary, if couplings exist, a solution procedure implying all subdomains must be provided to compute Lagrange Multipliers.

Existing approach consists in identifying in condensation operator **H** blocks that can be solved inside one subdomain, i. e. corresponding to groups of fully local links with only local nodes involved. Figure 4 illustrates the process in the elementary case of 2 subdomains. On each subdomain, dofs are partitioned between those concerned by any *remote* link (or a link coupled to a *remote* link, the two having to be treated simultaneously) and the others.



$$\mathbf{H}_{L1} = \mathbf{C}_{L1}\mathbf{M}_1^{-1}\mathbf{C}_{L1}^T$$

$$\mathbf{H}_{L2} = \mathbf{C}_{L2}\mathbf{M}_2^{-1}\mathbf{C}_{L2}^T$$

$$\mathbf{H}_{R} = \begin{bmatrix}\mathbf{C}_{R1} & \mathbf{C}_{R2}\end{bmatrix}\begin{bmatrix}\mathbf{M}_1^{-1} & \\ & \mathbf{M}_2^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{C}_{R1}^T \\ \mathbf{C}_{R2}^T\end{bmatrix}$$

Figure 3: Identification of diagonal blocks in condensation operator.

Global block $\mathbf{H}_R$ is then solved on a single processor, resulting in a loss of parallelism all the more important that kinematic couplings between subdomains are numerous. This can severely harm scalability, again for example when large displacements of structural fragments occur, these fragments interacting with initially distant fluid elements.

Let us also notice that building global block $\mathbf{H}_R$ requires centralizing all concerned *remote* links data on the processor dedicated to the solution. These communications done, a distributed linear solver is useless to compute Multipliers, since time necessary for data redistribution prior to solution easily compensate parallel acceleration provided by the solver.

3.2 Algorithmic solutions

Keeping the same distributed memory parallel structure for the program, two alternative parallel algorithms can be proposed to deal with the problematic exposed above. They are being developed and tested in the framework of French ANR REPDYN project [4]. In both cases, the point is to increase parallelism for the solution of the problem involving *remote* links.

3.2.1 Correct use of a distributed linear solver

As noticed earlier, this option is worthless if data have to be centralized on one unique processor to build to system to be solved. Instead, to achieve efficiency, a distributed linear solver must start from a matrix whose terms (either its lines or its terms individually) are already equally split upon available resources.

One way to provide such a distribution is to share among all processors data concerning *remote* links (and links coupled to them), yielding block $\mathbf{H}_R$ to be solved at global level. This represents a reduced extra-cost in communications, compared to existing transfer from all processors to one unique destination.

Each processor is then able to compute terms of $\mathbf{H}_R$. Its building becomes a parallel task and matrix distribution is controlled in order to benefit from a parallel solver, such as

MUMPS [5][6] for example.

3.2.2 Iterative solution with no operator building

Even if building global block becomes parallel and produces a distributed matrix, linear solver are known to suffer from limited scalability, especially in the present situation where the matrix size is small, the number of linked nodes being far lower than the total number of nodes.

An alternative strategy is to avoid building the operator within an iterative solution of the link problem, using Conjugate Gradient for example. Starting from system (3) with value of Lagrange Multipliers at iteration *k*, corresponding accelerations are obtained from (3-a):

$$\begin{bmatrix} \ddot{\mathbf{U}}_1^k \\ \ddot{\mathbf{U}}_2^k \\ \ddot{\mathbf{U}}_1^k \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1 & & \\ & \mathbf{M}_2 & \\ & & \mathbf{M}_3 \end{bmatrix}^{-1} \left(\begin{bmatrix} \mathbf{F}_{ext1} \\ \mathbf{F}_{ext2} \\ \mathbf{F}_{ext3} \end{bmatrix} - \begin{bmatrix} \mathbf{F}_{i1} \\ \mathbf{F}_{i2} \\ \mathbf{F}_{i3} \end{bmatrix} \text{-}\mathbf{C}^T \begin{bmatrix} \mathbf{\Lambda}_1^k \\ \mathbf{\Lambda}_1^k \\ \mathbf{\Lambda}_1^k \end{bmatrix} \right) \tag{4}$$

Residual on kinematic links is then computed by:

$$\mathbf{S}^k = \mathbf{C} \begin{bmatrix} \ddot{\mathbf{U}}_1^k \\ \ddot{\mathbf{U}}_2^k \\ \ddot{\mathbf{U}}_3^k \end{bmatrix} \text{-}\mathbf{B} \tag{5}$$

These two evaluations are parallel steps using operators local to subdomains. For residual computation, vector is obtained by blocks, each block corresponding to links written on a given subdomain.

Preconditioning the algorithm is necessary to speed-up convergence. One preconditioner of interest consists in solving links written on a subdomain, potentially involving dofs it does not own, as if they were not coupled to any other links on another subdomain, which the global link operator classically handles.

Considering again simplified case (4), this first means isolating in *remote* links block of connections matrix the links written locally:

$$\begin{bmatrix} \mathbf{C}_{\mathrm{R}1} & \mathbf{C}_{\mathrm{R}2} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{\mathrm{R}1}^1 & \mathbf{C}_{\mathrm{R}2}^1 \\ \mathbf{C}_{\mathrm{R}1}^2 & \mathbf{C}_{\mathrm{R}2}^2 \end{bmatrix} \tag{6}$$

Global condensation operator then writes:

$$\mathbf{H}_{\mathrm{R}} = \begin{bmatrix} \mathbf{C}_{\mathrm{R}1}^1 & \mathbf{C}_{\mathrm{R}2}^1 \\ \mathbf{C}_{\mathrm{R}1}^2 & \mathbf{C}_{\mathrm{R}2}^2 \end{bmatrix} \begin{bmatrix} \mathbf{M}_1^{-1} & \\ & \mathbf{M}_2^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{C}_{\mathrm{R}1}^{1\ T} & \mathbf{C}_{\mathrm{R}1}^{2\ T} \\ \mathbf{C}_{\mathrm{R}2}^{1\ T} & \mathbf{C}_{\mathrm{R}2}^{2\ T} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{\mathrm{R}}^1 & \mathbf{H}_{\mathrm{RC}} \\ \mathbf{H}_{\mathrm{RC}}^{\ T} & \mathbf{H}_{\mathrm{R}}^2 \end{bmatrix} \tag{6}$$

with
$$\mathbf{H}_{\mathrm{R}}^1 = \mathbf{C}_{\mathrm{R}1}^1 \mathbf{M}_1^{-1} \mathbf{C}_{\mathrm{R}1}^{1\ T} + \mathbf{C}_{\mathrm{R}2}^1 \mathbf{M}_2^{-1} \mathbf{C}_{\mathrm{R}2}^{1\ T}$$
$$\mathbf{H}_{\mathrm{R}}^2 = \mathbf{C}_{\mathrm{R}1}^2 \mathbf{M}_1^{-1} \mathbf{C}_{\mathrm{R}1}^{2\ T} + \mathbf{C}_{\mathrm{R}2}^2 \mathbf{M}_2^{-1} \mathbf{C}_{\mathrm{R}2}^{2\ T}$$
$$\mathbf{H}_{\mathrm{RC}} = \mathbf{C}_{\mathrm{R}1}^1 \mathbf{M}_1^{-1} \mathbf{C}_{\mathrm{R}1}^{2\ T} + \mathbf{C}_{\mathrm{R}2}^1 \mathbf{M}_2^{-1} \mathbf{C}_{\mathrm{R}2}^{2\ T}$$

Blocks $\mathbf{H}_R^1$ and $\mathbf{H}_R^2$ can be assembled locally on subdomains 1 and 2 respectively, knowing only locally written links and masses of all involved dofs, potentially belonging to another subdomain, which implies small communications.

Ignoring couplings in the preconditioner then consists in neglecting block $\mathbf{H}_{\mathrm{RC}}$ in the solution procedure. A diagonal per block operator is thus obtained, each block locally solved on a subdomain. Extension to any number of subdomains is straightforward.

4 TOWARDS HYBRID PARALLELISM

In addition to scalability improvement for solving kinematic links with domain decomposition, recent hardware evolution suggests an alternative parallelism: a distributed memory approch with domain decomposition between nodes interconnected by a high-performance network, coupled to a shared memory parallel acceleration using local cores inside a node, with optional additional acceleration obtained from GPU(s).

From a theoretical point of view, this allows to lighten concurrency between domain decomposition and kinematic links enforcement for a given number of processing units, provided an efficient multi-CPU/GPU shared memory parallelism.

Again, research in this area is carried out for EUROPLEXUS software in the framework of REPDYN project, in collaboration with MOAIS team from INRIA/LIG laboratory. Dynamic load balancing is handled through KAAPI library [7][8], implementing graph partitioning methods and work stealing between SMP threads.

A strict goal is to achieve a cooperative hybrid parallelism among all available resources efficient for all EUROPLEXUS functionalities. Originality is thus to propose different simulaneous parallel solutions adapting to multiple algorithms occurring together in a coupled simulation, instead of parallel efficiency demonstrations based on restricted model applications, which are classically hard to extend to industrial level.

REFERENCES

[1] F. Casadei, Fast Transient Fluid-Structure Interaction with Failure and Fragmentation, *8th World Congress on Computational Mechanics,* June 30-July 5, 2008.

[2] V. Faucher, S. Kokh. Explosive Fluid-Structure Interaction using Multi-Component Flows with Anti-Dissipation, *IV European Conference on Computational Mechanics,* May 16-21, 2010.

[3] F. Casadei, J.P. Halleux, A. Saha, F. Chille, Transient Fluid-Structure Interaction Algorithm for Large Industrial Applications, *Comp. Meth. in Appl. Mech. and Engrg.*, 2001.

[4] http://www.repdyn.fr

[5] P.R. Amestoy, I.S. Duff & J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers, *ENSEEIHT-IRIT Technical Report*, *1998. Revised version appeared in Comp. Meth. in Appl. Mech. Eng.*, **184**, 501-520 (2000).

[6] http://graal.ens-lyon.fr/MUMPS/

[7] T. Gautier, X. Besseron & L. Pigeon, KAAPI : a thread scheduling runtime system for data flow computations on cluster of multi-processors. *In Parallel Symbolic Computation'07 (PASCO'07),* number 15-23, London, Ontario, Canada, 2007.

[8] http://kaapi.gforge.inria.fr/