Available online at www.sciencedirect.com

SCIENCE DIRECT®

Comput. Methods Appl. Mech. Engrg. 195 (2006) 2028–2049

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Algorithms for strong coupling procedures

Hermann G. Matthies [*], Rainer Niekamp, Jan Steindorf

Institute of Scientific Computing, Technische Universität Braunschweig, D-38092 Brunswick, Germany

Received 1 April 2004; received in revised form 8 November 2004; accepted 15 November 2004

Abstract

This paper considers algorithms for computing the response of a coupled problem with a partitioned approach. One important example treated here is fluid–structure interaction (FSI). Often procedures or even software exists to solve each sub-problem separately, and one wants to couple both. This setting seems to allow only the so-called weak coupling which is not sufficient for some problems. The so-called strong coupling is often a totally implicit formulation, where the system components are evaluated at the same time level. This usually requires an iterative procedure. The FSI problem is cast as an abstract differential algebraic equation (DAE), for which the coupling procedures are developed. With the partitioned approach, one simple and frequently used computational procedure is similar to a block-Gauss–Seidel iteration. It is shown why this approach may experience difficulties, and how they may be circumvented with Newton-like methods still staying in the partitioned framework. The functional and software engineering requirements for the simulation interface are described and analysed. Some simple coupled example problems demonstrate how the proposed procedures work.

Keywords: Partitioned methods; Strong coupling; Block-Newton methods; Quasi-Newton methods; Fluid–structure interaction

1. Introduction

Fluid–structure interaction (FSI) problems arise in a number of scientifically interesting and technologically important settings [1]. These often show strong interplay between the fluid and the structure [1–7], e.g. in the design of aircraft [8,9] and in many other situations [10].

The FSI problem is viewed here as an example of an abstract class of coupled interaction problems. This has the advantage that algorithms developed in one setting may be used in other situations where coupled problems arise [11], e.g. thermo-mechanical [12] coupling, or soil–pore fluid interaction [13], to name but a

[*] Corresponding author. Fax: +49 531 391 3003.
E-mail address: wire@tu-bs.de (H.G. Matthies).

0045-7825/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2004.11.032

few. There are different ways to tackle a coupled interaction problem, cf. [14–17]. One possibility is to develop new software and solution methods for each of these coupled applications, as without doubt will happen in some areas. This is referred to as a *monolithical* approach [18], or sometimes as the *direct* method [19]. On the other hand, one may assume that the methods and software systems which have been developed for either application will continue to be used, in this instance the fluid or structural software. Therefore *partitioned* methods [20,21,14,6,22,23] are considered, also known as *iterative* methods [19] for fluid–structure interaction, i.e. separate solvers are used for the fluid and the structure [24,25]. In [28–30] one may find descriptions of such explicit staggering schemes, in [28] they are analysed in terms of their order of consistency in the coupling. They may be designed serially or in parallel [24,28]. Here we follow [25,26] and formulate the FSI problem in an abstract setting as a system of differential algebraic evolution equations (DAEs), coupled by an algebraic condition. The term algebraic is to be understood in the way that the coupling has no time evolution of its own, and will only involve differential operators in space.

For stability reasons, often a fully implicit formulation has to be used [7,19,30]. In this approach, one has to solve a large system of nonlinear equations, preferably with the use of the (iterative) solvers for the subsystems. Commonly this is performed with block-Jacobi, block-Gauss–Seidel or related relaxation methods [31,30].

These simple methods sometimes fail [14,32,25,33–36] rather unexpectedly. We will introduce here superior approximative Newton-like methods [37,25,33–36], which are not plagued by such occasional failures, but still retain the partitioned approach on a software level.

Apart from the FSI setting, these ideas can be applied in a multitude of other areas such as general multi-field/multi-physics phenomena [14,9], co-simulation and multi-numerics [38,39,32,40], domain-decomposition [41], wave-form relaxation [42], and are in line with efforts of parallelisation and modularisation of software for distributed simulation.

Algorithms derived from Newton's method have been considered in [43,44] for such coupled problems. Certain parts of the Jacobian of the implicit iteration may not be accessible explicitly [37,38,25,26], making the direct application of Newton's method difficult. Luckily, the Jacobian is not needed explicitly here, one only has to solve linear equations with it as system matrix. In a *matrix-free approach* with an inner Krylov-type linear solver only the action of the matrix on a vector is needed. In this case the action of the missing parts of the Jacobian can be suitably approximated. In some instances it also helps not to solve the equilibrium equations, but an equivalent, numerically better conditioned system [37,38,25,33–36,26]. This subject will be taken up later in the paper.

The plan of the paper is as follows. In Section 2 the example of a FSI problem is introduced in a simple way which adopts the ALE-setting [4]. In Section 3 an abstract setting of coupled subsystems is described following [25,26]. The time evolution is assumed to be given by an abstract ordinary differential equation for each subsystem. One may distinguish between pure differential coupling, and a coupling by an "algebraic" constraint or coupling condition. In the latter case it does not add much complexity to allow the subsystems to be modelled by DAEs. Actually for the incompressible Navier–Stokes equation the incompressibility constraint is like an "algebraic" constraint. Upon time discretisation by some finite difference method, one distinguishes between explicit and implicit methods, where this refers to the total system. It is the implicit methods that require solutions over the global system.

In Section 4 the FSI formulation is shown to fit the abstract setting after some manipulation. The solution of the global system is first addressed in Section 5, where nonlinear block-Jacobi and block-Gauss–Seidel methods are an obvious possibility. Their occasional failure is described and explained, and a superior family of block-Newton-like methods is proposed. As pointed out already, the difficulty here is that parts of the Jacobian of the global system, including the cross-derivatives, are not always accessible explicitly. Some recipes for treating this question are given in Section 5.

Sections 6 and 7 give some view behind the used software-architecture, which is based on the definition of a simulation interface and an abstract communication middle-ware called *Component Template Library*

(CTL). This architecture suggests a standardisation of simulation codes in order to yield an optimal re-usability for partitioned simulation. The requirements on the various solvers described in Section 5 for the global iteration are also given.

A simple structure–structure coupling and a simple FSI example in Section 8 show the effectivity of the proposed methods. The final Section 9 concludes with an outlook on similar problems and possible future work.

2. Description of fluid–structure interaction

Before describing in the abstract setting of the following Section 3 how to solve coupled systems numerically by using the subsystem solvers, the fluid–structure interaction (FSI) problem is formulated. It includes the necessary modifications due to the fact that the fluid is usually described in an Eulerian setting, whereas the structure is commonly described in a Lagrangian formulation. These have to be made to fit, at least at the common interface, and therefore the fluid formulation is in an arbitrary Lagrangian–Eulerian (ALE) frame. In this way parts of the fluid domain may change in time, and at the interface to the structure they can fit the actual displacement (Lagrangian frame) [4,47].

2.1. The fluid

Assume the fluid to be incompressible Newtonian, and satisfying the appropriate Navier–Stokes equation. Allowing compressibility or non-Newtonian constitutive models would change the details of the fluid movement, but not the abstract coupling formulation in Section 3, nor the algorithms derived from it. However it is the example of an incompressible fluid which makes it more challenging in one respect, as the incompressibility condition is an "algebraic" constraint. The Navier–Stokes equation in an arbitrary Lagrangian–Eulerian (ALE) [4,47,48] framework in the moving spatial fluid domain Ω_{f} is

$$\varrho_{\mathrm{f}}(\dot{v} + ((v - \dot{\mathscr{X}}) \cdot \nabla)v) - \operatorname{div}\sigma + \nabla p = r_{\mathrm{f}}, \tag{1}$$

$$2\sigma = \nu(\nabla v + (\nabla v)^{\mathrm{T}}), \quad \operatorname{div} v = 0. \tag{2}$$

The boundary $\partial\Omega_{\mathrm{f}}$ we assume to be divided into three disjoint parts $\partial\Omega_{\mathrm{f}} = \Gamma_{\mathrm{v}} \cup \Gamma_{\mathrm{q}} \cup \Gamma_{\mathrm{c}}$ where on Γ_{v} the velocity is prescribed, on Γ_{q} the traction is given, and Γ_{c} is the coupling boundary, where the coupling conditions will be specified below. As the first two kinds of boundary conditions are completely standard, their detailed description is skipped. Here ϱ_{f} is the fluid density, v the velocity, and the superimposed dot denotes the time derivative $\dot{v} := \partial_t v = \partial v/\partial t$. The viscous stress is σ, $\mathscr{X}$ is the position of the reference ALE-coordinate system, and $\dot{\mathscr{X}}$ its velocity. The fluid shear viscosity is denoted by ν, p is the pressure, r_{f} the body force in the fluid, and the differential operators are in the spatial frame.

The movement of the ALE-coordinate system $\mathscr{X}(t)$ is not specified yet, this will be done after considering the discretisation.

2.2. The structure

The structure is described in a Lagrangian framework in a fixed material domain Ω_{s}. For the sake of simplicity it is assumed that the constitutive model is a St. Venant material, hence the equilibrium equation takes the following form:

$$\varrho_{\mathrm{s}}\ddot{u} - \mathrm{DIV}(FS) = r_{\mathrm{s}}, \quad F = I + \mathrm{GRAD}\, u, \tag{3}$$

$$S = \lambda(\operatorname{tr} E)I + 2\mu E, \quad 2E = (C - I), \quad C = F^{\mathrm{T}}F. \tag{4}$$

The boundary $\partial\Omega_s$ is generally divided into three disjoint parts $\partial\Omega_s = \Gamma_u \cup \Gamma_t \cup \Gamma_c$, where on Γ_u the displacements u are prescribed, on Γ_t the tractions, and Γ_c is the coupling boundary with the fluid. The first two boundary conditions are again completely standard, hence their description is again skipped. The other quantities in Eq. (3) are the structure density ϱ_s, the displacement gradient F, the second Piola–Kirchhoff stress S, and the body load r_s. The Lamé moduli are denoted by λ and μ, and E is the Lagrange–Green strain, derived from the Cauchy–Green tensor C, and the capitalised differential operators are in the material frame.

Again here it would be possible to assume other models for the solid, e.g. non-elastic ones with internal variables such as plasticity or damage, they would still fit into the general setting to be introduced in Section 3.

2.3. The interface

On the coupling boundary Γ_c let a unique normal n be defined in the spatial frame. Supposing a vanishing initial displacement $u(\mathscr{X}_0, t_0) = 0$, where $\mathscr{X}_0 = \mathscr{X}(t_0)$, the coupling conditions at a later time t may be expressed by coincident velocities of the fluid v and structure $\dot{u}$ at the spatial location $\mathscr{X}(t) = \mathscr{X}_0 + u(\mathscr{X}_0, t)$:

$$v(\mathscr{X}(t), t) = \dot{u}(\mathscr{X}_0, t). \tag{5}$$

Eq. (5) ensures in this strong formulation also the coincidence of the positions at time $t > t_0$.

Additionally the tractions from fluid and structure have to balance each other:

$$(\sigma - pI) \cdot n = -\frac{1}{J} FSF^{\mathrm{T}} \cdot n, \quad J = \det F. \tag{6}$$

3. Abstract coupled systems

In this section a general abstract formulation for coupled systems is presented, and in the following Section 4 it will be shown how the FSI problem fits this abstract setting. In order to keep the formulation simple, only a global system composed of two subsystems will be considered. It is no problem to extend the ideas and methods presented here to more than two coupled subsystems, but this makes it more difficult to convey the basic ideas, which may be seen already in the simplest case of two subsystems.

The case of pure differential coupling is a good point to start in Section 3.1. This is a situation where some variables in both subsystems may be directly identified. Often the coupling will introduce additional variables, as will be seen in the FSI example, together with algebraic equations in addition to the subsystem equations. It leads to systems of coupled DAEs in Section 3.2.

3.1. Pure differential coupling

Assume that the time evolution of the first subsystem is given by a differential equation of the form

$$\dot{x}_1 = f_1(x_1, x_2), \tag{7}$$

where $x_1 \in \mathscr{X}_1$, $x_2 \in \mathscr{X}_2$ are elements of some Banach-spaces. The spaces $\mathscr{X}_1$ and $\mathscr{X}_2$ may be spaces of functions, and f_1 may be a differential operator, so that Eq. (7) may describe subsystems modelled by partial differential equations, such as fluid flow. The variables x_2 are not further specified in this model, they provide the coupling to the second subsystem, which is described completely analogous to Eq. (7):

$$\dot{x}_2 = f_2(x_2, x_1). \tag{8}$$

The coupled system equations (7) and (8) are nothing but a larger evolution system for the combined vector $(x_1, x_2) \in \mathscr{X}_1 \times \mathscr{X}_2$, and actually may come from splitting a larger system into smaller ones for parallelisation, such as in wave-form relaxation [42]. All variables have differential evolution equations, and therefore we refer to this case as pure differential coupling. In a later Section 3.2 the case when the coupling is modelled by "algebraic" equations will be considered, and the subsystems themselves may be DAEs.

With this simple set-up already a number of solution concepts may be defined which have been used frequently (e.g. [16]) in such coupled simulations, but which also occur in other contexts as with an artificial splitting for computational purposes [42]. In [28] is a recent overview over several such coupling procedures, see also [30] for a "closed system" view of the coupling problem.

3.1.1. *Explicit coupling*

Assume that both subsystem equations (7) and (8) have been discretised in time with some appropriate method. If the methods for the individual subsystems are both explicit, it is probably most natural to stay explicit also for the coupled system. For the sake of simplicity of exposition assume that both subsystems have the same time step Δt. Otherwise one subsystem may be sub-cycling. Then what is referred to as one step for that system here is really a number of steps, and one considers the interval between synchronisation points. Denoting the discrete approximation to the solution of Eqs. (7) and (8) at step n by $x_\jmath^{(n)}$, $(\jmath = 1, 2)$, the explicit integration algorithms for the two subsystems may be expressed as

$$x_1^{(n)} = \varphi_1(x_1^{(n-1)}, y_2), \tag{9}$$

$$x_2^{(n)} = \varphi_2(x_2^{(n-1)}, y_1), \tag{10}$$

where dependence on time t or the time step Δt has been omitted, and the two functions $y_\jmath(t)$, $(\jmath = 1, 2)$ are assumed given. They are to represent the variables of the other subsystem in the time interval of length $\Delta t = t_n - t_{n-1}$, but of course these are not yet known. Staying explicit, one may extrapolate these by some function $\Psi_\jmath(x_\jmath^{(n-1)})$, $(\jmath = 1, 2)$ of the past values of the approximate solution, in the simplest case one may assume the value at t_{n-1} as constant throughout the interval, i.e. $y_\jmath(t) = \Psi_\jmath(x_\jmath^{(n-1)}) \equiv x_\jmath^{(n-1)}$, $(\jmath = 1, 2)$. This leads to the simplest case of *weak* or *loose* coupling, the so-called *staggering* method [22,45,28], here with explicit subsystem integrators:

$$x_1^{(n)} = \varphi_1(x_1^{(n-1)}, \Psi_2(x_2^{(n-1)})) = \varphi_1(x_1^{(n-1)}, x_2^{(n-1)}), \tag{11}$$

$$x_2^{(n)} = \varphi_2(x_2^{(n-1)}, \Psi_1(x_1^{(n-1)})) = \varphi_2(x_2^{(n-1)}, x_1^{(n-1)}). \tag{12}$$

With this, the method is completely specified. The single system integrators are most likely going to have an associated critical time step $\Delta t_{c\jmath}$, $(\jmath = 1, 2)$, and by coupling the systems in this explicit fashion the critical time step for the global system will be not more than $\min_\jmath \Delta t_{c\jmath}$. The two sub-steps in Eqs. (11) and (12) can be performed in parallel. If some of this inherent parallelism is sacrificed, a partly implicit method may be formulated: First perform Eq. (11), to be followed by

$$x_2^{(n)} = \varphi_2(x_2^{(n-1)}, \Psi_1(x_1^{(n-1)})) = \varphi_2(x_2^{(n-1)}, x_1^{(n)}), \tag{13}$$

where the new value

$$x_1^{(n)} = \Psi_1(x_1^{(n-1)}) := \varphi_1(x_1^{(n-1)}, x_2^{(n-1)}) \tag{14}$$

is already used. This is how the staggering method is mostly applied. Of course the combined method is still explicit, but only "half as much" as before. It is sometimes called the "Conventional Serial Staggered (CSS)" scheme [28].

In case that the subsystem integrators are implicit, they may still be combined in an explicit fashion [26]. For the system as a whole this is still explicit, and will again introduce limitations connected with a critical time step associated to the global system [30]. To circumvent this, generally one has to introduce a global coupling, although there are methods to retain at least the linear stability characteristics in the staggered approach [21,14,46].

3.1.2. Implicit coupling

To alleviate the time step restriction just mentioned, one may formulate the whole system in an implicit way, extrapolating $y_\jmath(t) = \Psi_\jmath(x_\jmath^{(n-1)}) \equiv x_\jmath^{(n)}$, $(\jmath = 1,2)$ with the constant and yet unknown value of the approximate solution at the end of the time step. Again here some more elaborate approximation is possible, but we restrict ourselves to this simplest and most used case for the sake of brevity. The equation system to be solved at each time step now reads

$$x_1^{(n)} = \phi_1(x_1^{(n)}, x_1^{(n-1)}, \Psi_2(x_2^{(n-1)})) = \phi_1(x_1^{(n)}, x_1^{(n-1)}, x_2^{(n)}), \tag{15}$$

$$x_2^{(n)} = \phi_2(x_2^{(n)}, x_2^{(n-1)}, \Psi_1(x_1^{(n-1)})) = \phi_2(x_2^{(n)}, x_2^{(n-1)}, x_1^{(n)}). \tag{16}$$

This is a case of *strong* or *tight* coupling, and the results from this are completely equivalent to what would be achieved by a monolithical formulation, although this here is the result of a partitioned approach. Eqs. (15) and (16) are a coupled system, and cannot be solved independently of each other. Hence, even if methods to solve each equation separately are available, this does not solve the system equations (15) and (16), a global iteration is needed.

Thus although the formulation of the *partitioned* but *strongly coupled* approach is equivalent to a *monolithical* approach, the methods to solve the coupled system will be different: In the monolithical approach all the information desired about the subsystems is at disposal, and the solution methods are sometimes termed as *direct* [19], although the actual solution process may very well be iterative.

In the partitioned approach, not all the information desired about the subsystems is available, and there has to be some kind of iteration across the subsystems. Therefore this approach is sometimes termed the *iterative* [19] method.

3.2. Differential and algebraic coupling

Eqs. (15) and (16) in the last Section 3.1.2 are a global system to be solved at each time step. It will turn out that the case of two coupled differential algebraic equations (DAEs) is no more difficult conceptually, so this situation is considered next [32].

Assume that the two subsystems are DAEs of index 1, denoted by

$$\dot{x}_1 = f_1(x_1, x_2, y_1, y_2, z), \tag{17}$$

$$0 = g_1(x_1, x_2, y_1, y_2, z). \tag{18}$$

The description is similar to Eq. (7), only that now there is additionally a local algebraic variable $y_1 \in \mathscr{Y}_1$ and a global one $z \in \mathscr{Z}$. Again $\mathscr{Y}_1$ and $\mathscr{Z}$ are some suitable Banach spaces. As with the evolution law f_1 in Eq. (17), also the function g_1 in the "algebraic" Eq. (18) may be a spatial differential operator. Completely analogous, just by switching indices, we model the second subsystem by

$$\dot{x}_2 = f_2(x_2, x_1, y_2, y_1, z), \tag{19}$$

$$0 = g_2(x_2, x_1, y_2, y_1, z), \tag{20}$$

with $y_2 \in \mathscr{Y}_2$ in some appropriate Banach space $\mathscr{Y}_2$.

The two subsystems are now not only coupled by the variables $(x_\jmath, y_\jmath)$ $(\jmath = 1,2)$ as before, but one may additionally allow for a global algebraic coupling

$$0 = h(x_1, x_2, y_1, y_2, z), \tag{21}$$

through the global algebraic variable $z \in \mathscr{Z}$.

The assumption that each subsystem, the system equations (17), (18) and (21) with x_2, y_2 given as known functions, the system equations (19), (20) with x_1, y_1 given as known functions, and also the global system is a DAE of index 1 is equivalent to the condition that the operator matrices

$$D_{y_j} g_j, \quad \begin{bmatrix} D_{y_j} g_j & D_z g_j \\ D_{y_j} h & D_z h \end{bmatrix}, \quad (j = 1, 2), \quad \begin{bmatrix} D_y g & D_z g \\ D_y h & D_z h \end{bmatrix} \tag{22}$$

be regular, where D_q is the partial derivative w.r.t. q, and $g = (g_1, g_2)^T$ and $y = (y_1, y_2)^T$. These matrices are sometimes also called coupling matrices.

3.2.1. Implicit DAE coupling

Differential algebraic equations are usually solved with implicit methods, assume that here the algebraic constraint is satisfied in each step, this will also be followed here. Upon discretising Eqs. (17), (18) and Eqs. (19), (20) in time with an implicit method, one arrives at an equation to be solved for each subsystem.

Treating also the global algebraic condition Eq. (21) implicitly, the global system to be solved is:

$$x_1^{(n)} = \Phi_1(x_1^{(n)}, x_1^{(n-1)}, x_2^{(n)}, y_1^{(n)}, y_2^{(n)}, z^{(n)}), \tag{23}$$

$$x_2^{(n)} = \Phi_2(x_2^{(n)}, x_2^{(n-1)}, x_1^{(n)}, y_2^{(n)}, y_1^{(n)}, z^{(n)}), \tag{24}$$

$$0 = h(x_1^{(n)}, x_2^{(n)}, y_1^{(n)}, y_2^{(n)}, z^{(n)}). \tag{25}$$

The last Eq. (25) is usually subsumed with one of the subsystems Eq. (23) or Eq. (24). This fits into the framework of Section 3.1.2, and the situation is as described by Eqs. (15) and (16), in that a global system has to be solved, and only solvers for the subsystem are available.

4. Reformulation of FSI as a coupled DAE

Assume now that the fluid as described by Eqs. (1) and (2), the structure given by Eqs. (3) and (4), and the interface specified by Eqs. (5) and (6) have been discretised in space. Everyone may use her favourite discretisation, e.g. finite elements, finite volumes etc., e.g. [11,49,50]. The space-discretised variables from those equations are denoted by the corresponding bold-face letter.

4.1. Semi-discrete form of FSI

The movement of the reference coordinate system in the fluid domain still has to be detailed. Several possibilities exist [51–53,7], here the connections between the nodes in the fluid domain are modelled as elastic springs [54]. This fictitious inertia-free elastic body has displacement loading from the moving structure, on the outer boundaries it is fixed. The traction balance Eq. (6) introduces additional forces $\boldsymbol{T}_f^T \boldsymbol{\tau}$ on the fluid, and the reaction force $\boldsymbol{T}_s^T \boldsymbol{\tau}$ on the solid, where $\boldsymbol{\tau}$ is the discretised traction vector on the interface Γ_c. Hence for the fluid the complete semi-discrete equations of motion are [25]:

$$\boldsymbol{M}_f \dot{\boldsymbol{v}} + \boldsymbol{N}_f(\boldsymbol{v} - \dot{\boldsymbol{\mathscr{X}}})\boldsymbol{v} + \boldsymbol{K}_f \boldsymbol{v} + \boldsymbol{B}_f \boldsymbol{p} = \boldsymbol{r}_f + \boldsymbol{T}_f^T \boldsymbol{\tau}, \tag{26}$$

$$\boldsymbol{B}_f^T \boldsymbol{v} = \boldsymbol{0}, \tag{27}$$

$$\boldsymbol{K}_g \boldsymbol{\mathscr{X}} = \boldsymbol{A}\boldsymbol{u}. \tag{28}$$

The terms in Eq. (26) are the discrete analogues of those in Eq. (1), the term $\boldsymbol{r}_{\rm f}$ includes the prescribed boundary stresses, $\boldsymbol{M}_{\rm f}$ is the mass matrix, the term involving $\boldsymbol{N}_{\rm f}$ a nonlinear contribution from the convective acceleration, $\boldsymbol{K}_{\rm f}$ the matrix of the viscous term, and $\boldsymbol{B}_{\rm f}$ a discrete form of the gradient. In Eq. (27) we may recognise the discrete form of the incompressibility condition, and Eq. (28) describes the movement $\mathscr{X}(t)$ of the fluid domain, driven by the structure displacements $\boldsymbol{u}$ transferred to the common interface $\Gamma_{\rm c}$ by the transfer matrix $\boldsymbol{A}$, and the grid stiffness matrix $\boldsymbol{K}_{\rm g}$ is due to the springs just mentioned between the nodes in the fluid domain.

In a similar vein for the structure one obtains

$$\boldsymbol{M}_{\rm s}\ddot{\boldsymbol{u}} + \boldsymbol{K}_{\rm s}(\boldsymbol{u})\boldsymbol{u} = \boldsymbol{r}_{\rm s} - \boldsymbol{T}_{\rm s}^{\rm T}\boldsymbol{\tau}. \tag{29}$$

Here $\boldsymbol{M}_{\rm s}$ is the mass matrix, $\boldsymbol{K}_{\rm s}$ the possibly displacement dependent stiffness matrix, $\boldsymbol{r}_{\rm s}$ represents the body and traction forces, and $\boldsymbol{T}_{\rm s}^{\rm T}\boldsymbol{\tau}$ is due to the forces of the fluid on the structure on the interface $\Gamma_{\rm c}$ as given by Eq. (6). The coupling condition for the velocities Eq. (5) in discrete form is

$$\boldsymbol{T}_{\rm f}\boldsymbol{v} = \boldsymbol{T}_{\rm s}\dot{\boldsymbol{u}}, \tag{30}$$

where due to the variational formulation of the discretisation of Eq. (5) the transposes of the transfer matrices in Eqs. (26) and (29) appear.

This combined set, Eqs. (26)–(28) and Eqs. (29), (30), is a system of 2nd order differential algebraic equations (DAEs) of index 2 in the time variable t, as can be easily verified. In order to allow for an easier numerical treatment, and to make them fit the general formulation in Section 3.2, these equations will be converted to index 1 by differentiation [25].

There are quite a few items we have glossed over in our quick description, as these are not central to our issue. Nevertheless, for a proper implementation of FSI they have to be dealt with. This includes the problem of non-matching grids for fluid, structure, and possibly interface and the ALE-fluid domain. There are different possibilities to tackle those problems, such as consistent interpolation and mortar elements, to name but a few [55–58]. Another problem is a kind of consistency when moving the ALE-fluid mesh; the numerical time stepping algorithm together with the spatial discretisation has to satisfy a property commonly referred to as the geometric conservation law (GCL) [51,59].

4.2. Index and order reduction

As the fluid by itself is a DAE of index 2, the index will be reduced by differentiation. Differentiating Eq. (27) once gives $\boldsymbol{B}_{\rm f}^{\rm T}\dot{\boldsymbol{v}} = \boldsymbol{0}$. As $\boldsymbol{M}_{\rm f}$ is non-singular, one may solve for $\dot{\boldsymbol{v}}$ from Eq. (26) and insert this into the above relation, giving

$$\boldsymbol{B}_{\rm f}^{\rm T}\boldsymbol{M}_{\rm f}^{-1}(\boldsymbol{r}_{\rm f} + \boldsymbol{T}_{\rm f}^{\rm T}\boldsymbol{\tau} - \boldsymbol{N}_{\rm f}(\boldsymbol{v}-\boldsymbol{\psi})\boldsymbol{v} - \boldsymbol{K}_{\rm f}\boldsymbol{v} - \boldsymbol{B}_{\rm f}\boldsymbol{p}) = \boldsymbol{0}, \tag{31}$$

where $\boldsymbol{\psi} := \dot{\mathscr{X}}$ is the grid velocity. In the same manner Eq. (28) for the grid movement is differentiated

$$\boldsymbol{K}_{\rm g}\boldsymbol{\psi} - \boldsymbol{A}\boldsymbol{w} = \boldsymbol{0}, \tag{32}$$

where the new variables $\boldsymbol{w} := \dot{\boldsymbol{u}}$, the structural velocities have been introduced.

With this relation the structural equation (29) can be reduced to a first order system in a standard manner:

$$\dot{\boldsymbol{u}} = \boldsymbol{w}, \tag{33}$$

$$\boldsymbol{M}_{\rm s}\dot{\boldsymbol{w}} + \boldsymbol{K}_{\rm s}(\boldsymbol{u})\boldsymbol{u} = \boldsymbol{r}_{\rm s} - \boldsymbol{T}_{\rm s}^{\rm T}\boldsymbol{\tau}. \tag{34}$$

This index reduction may cause a drift-off from the algebraic constraint. Therefore stabilisation techniques must be used, see [63].

4.3. The DAE correspondence

For the concrete application, the abstract equations (17) and (18) may now be identified with the following terms [25]:

$$\boldsymbol{x}_1 = \begin{bmatrix} \boldsymbol{v} \\ \mathscr{X} \end{bmatrix}, \quad \boldsymbol{y}_1 = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{p} \\ \boldsymbol{\psi} \end{bmatrix}, \quad \boldsymbol{f}_1 = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{\psi} \end{bmatrix}, \quad \boldsymbol{z} = \boldsymbol{\tau}, \tag{35}$$

$$\boldsymbol{g}_1 = \begin{bmatrix} \boldsymbol{M}_\mathrm{f}\boldsymbol{b} + \boldsymbol{N}_\mathrm{f}(\boldsymbol{v}-\boldsymbol{\psi})\boldsymbol{v} + \boldsymbol{K}_\mathrm{f}\boldsymbol{v} + \boldsymbol{B}_\mathrm{f}\boldsymbol{p} - \boldsymbol{r}_\mathrm{f} - \boldsymbol{T}_\mathrm{f}^\mathrm{T}\boldsymbol{\tau} \\ -\boldsymbol{B}_\mathrm{f}^\mathrm{T}\boldsymbol{M}_\mathrm{f}^{-1}(-\boldsymbol{N}_\mathrm{f}(\boldsymbol{v}-\boldsymbol{\psi})\boldsymbol{v} - \boldsymbol{B}_\mathrm{f}\boldsymbol{p} - \boldsymbol{K}_\mathrm{f}\boldsymbol{v} + \boldsymbol{r}_\mathrm{f} + \boldsymbol{T}_\mathrm{f}^\mathrm{T}\boldsymbol{\tau}) \\ \boldsymbol{K}_\mathrm{g}\boldsymbol{\psi} - \boldsymbol{A}\boldsymbol{w} \end{bmatrix}; \tag{36}$$

so the first subsystem is the fluid with the ALE-grid. In Eq. (35,2) the new variable $\boldsymbol{b} := \dot{\boldsymbol{v}}$, the fluid acceleration, has been introduced.

The second subsystem is the structure,

$$\boldsymbol{x}_2 = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{w} \end{bmatrix}, \quad \boldsymbol{y}_2 = \boldsymbol{a}, \quad \boldsymbol{f}_2 = \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{a} \end{bmatrix}, \quad \boldsymbol{z} = \boldsymbol{\tau}, \tag{37}$$

$$\boldsymbol{g}_2 = \boldsymbol{M}_\mathrm{s}\boldsymbol{a} + \boldsymbol{K}_\mathrm{s}(\boldsymbol{u})\boldsymbol{u} - \boldsymbol{r}_\mathrm{s} + \boldsymbol{T}_\mathrm{s}^\mathrm{T}\boldsymbol{\tau}; \tag{38}$$

where no index reduction, but only a standard order reduction of the differential equation was needed. In Eq. (37,2) the new variable $\boldsymbol{a} := \dot{\boldsymbol{w}} = \ddot{\boldsymbol{u}}$, the structural acceleration, has been introduced.

The final item to identify is the global coupling condition,

$$\boldsymbol{h} = \boldsymbol{T}_\mathrm{f}\boldsymbol{b} - \boldsymbol{T}_\mathrm{s}\boldsymbol{a}, \tag{39}$$

which expresses the equality of the accelerations at the fluid–structure interface.

To verify the index 1 conditions Eq. (22) is a lengthy calculation [25], but the conditions are indeed satisfied as long as $\boldsymbol{K}_\mathrm{g}$, $\boldsymbol{M}_\mathrm{f}$, and $\boldsymbol{M}_\mathrm{s}$ are regular, here they can even be assumed to be symmetric positive definite.

5. Numerical procedures for partitioned methods

The computational procedures are very dependent on the formulation chosen for the global system. If, as was done here, the coupled system is formulated as a differential–algebraic equation, one is almost inevitably led to an at least partly implicit formulation, and such mixed time-discretisations have also been investigated [60,61]. In the present example this already appears within the fluid subsystem. The incompressibility condition is an algebraic constraint, and practically all discretisations have some implicit elements [50].

The equations describing the system are

- *the fluid problem* with the components
  1. the Navier–Stokes Eq. (1), or its semi-discrete form Eq. (26) as an evolution equation.
  2. the incompressibility condition Eq. (2), or its discrete form Eq. (27) as algebraic constraint.
  3. the ALE-grid movement, here only formulated in discrete form as an additional algebraic constraint Eq. (28).
- *the structure problem* in Eq. (3), or its semi-discrete form Eq. (29) as an evolution equation.

- *the coupling condition* Eqs. (5) and (6) on the spatial interface Γ_c between the fluid and the structural domain, or its discrete form Eq. (30) as an algebraic constraint.

The grid movement could very well be taken as a subsystem in its own right, but because it is so intimately connected to the fluid problem, it is customarily treated simultaneously with the fluid.

Through index and order reduction the above problems have been reduced to the abstract coupled DAE system equations (35)–(38). Assume that at least both subsystems have some implicit parts, something which is true for the example and also in line with common numerical wisdom for the discretisation of DAEs [63]. In the present partitioned and modular approach this also means that one has solvers for each subsystem, written here in fixed point form (see Eqs. (44) and (45)), as they may be used during the iterative solution process for each subsystem. This certainly includes also direct solvers, they may be simply regarded as iterative solvers that only need one iteration.

After discretisation in time, one has to solve Eqs. (23)–(25). The first Eq. (23) solves for the variables $(\boldsymbol{x}_1, \boldsymbol{y}_1)^{\mathrm{T}} = (\boldsymbol{v}, \mathscr{X}, \boldsymbol{b}, \boldsymbol{p}, \boldsymbol{\psi})^{\mathrm{T}}$ assuming the others are given, the second Eq. (24) solves for $(\boldsymbol{x}_2, \boldsymbol{y}_2)^{\mathrm{T}} = (\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{a})^{\mathrm{T}}$ assuming the others are given, and the last Eq. (25) solves for $\boldsymbol{z} = \boldsymbol{\tau}$ assuming the others are given. For simplicity, and as it is also customarily done that way, in the solution process the discrete coupling condition is included with one of the other two subsystems, i.e. either with the fluid, meaning that on the coupling interface the velocities are prescribed (Dirichlet data), and the interface tractions are passed from the fluid to the structure (Neumann data); or the other way around. In the first case for the concise formulation of the iteration denote the variables as

$$\xi := (\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{z})^{\mathrm{T}} = (\boldsymbol{v}, \mathscr{X}, \boldsymbol{b}, \boldsymbol{p}, \boldsymbol{\psi}, \boldsymbol{\tau})^{\mathrm{T}}, \tag{40}$$

$$\zeta := (\boldsymbol{x}_2, \boldsymbol{y}_2)^{\mathrm{T}} = (\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{a})^{\mathrm{T}}. \tag{41}$$

In the second case with Neumann data on the fluid and Dirichlet data on the structure the variable $\boldsymbol{z} = \boldsymbol{\tau}$ is included in ζ and not in ξ:

$$\xi := (\boldsymbol{x}_1, \boldsymbol{y}_1)^{\mathrm{T}} = (\boldsymbol{v}, \mathscr{X}, \boldsymbol{b}, \boldsymbol{p}, \boldsymbol{\psi})^{\mathrm{T}}, \tag{42}$$

$$\zeta := (\boldsymbol{x}_2, \boldsymbol{y}_2, \boldsymbol{z})^{\mathrm{T}} = (\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{\tau})^{\mathrm{T}}. \tag{43}$$

In order to concentrate on how to solve the coupled Eqs. (23)–(25), the time step counter is dropped, and also the dependence on the previous time steps is suppressed. One may assume both iterative solution processes are in fixed point form, i.e. the iteration

$$\xi_\kappa = \boldsymbol{F}_1(\xi_{\kappa-1}, \zeta), \quad \kappa = 1, 2, \ldots \tag{44}$$

converges for reasonable starting values ξ_0 with given ζ. Similarly, the iteration process

$$\zeta_\kappa = \boldsymbol{F}_2(\zeta_{\kappa-1}, \xi), \quad \kappa = 1, 2, \ldots \tag{45}$$

converges for reasonable starting values ζ_0 with given ξ.

Other approaches may start directly from the equilibrium equation, which for a unified description may also be assumed to be in fixed point form like Eqs. (44) and (45).

5.1. Non-linear block-Jacobi

The conceptually simplest way to use these two solvers F_1 and F_2 to solve the combined system is a non-linear block-Jacobi process, or in the context of domain decomposition it would be called an additive or parallel Schwarz procedure [41]. Given $\xi_{\kappa-1}$ and $\zeta_{\kappa-1}$, perform the following iterative step:

$$\xi_\kappa = \boldsymbol{F}_1^{\nu_1}(\xi_{\kappa-1}, \zeta_{\kappa-1}), \tag{46}$$

$$\zeta_\kappa = \boldsymbol{F}_2^{\nu_2}(\zeta_{\kappa-1}, \xi_{\kappa-1}) \tag{47}$$

meaning that in Eq. (46) the iteration Eq. (44) has been performed v_1 times with a fixed $\boldsymbol{\zeta}_{\kappa-1}$, and in Eq. (47) the iteration Eq. (45) has been performed v_2 times with a fixed $\boldsymbol{\xi}_{\kappa-1}$. Very often one takes $v_1 = v_2 = 1$. The starting values $\boldsymbol{\xi}_0$ and $\boldsymbol{\zeta}_0$ are provided by the extrapolation mapping Eq. (14). One iteration here is similar to one step in Eqs. (9) and (10) in the purely explicit coupling in Section 3.1.1.

5.2. Non-linear block-Gauss–Seidel

It is well known, that usually the corresponding Gauss–Seidel process converges faster [31], and so we are led to: Given $\boldsymbol{\xi}_{\kappa-1}$ and $\boldsymbol{\zeta}_{\kappa-1}$, do

$$\boldsymbol{\xi}_\kappa = \boldsymbol{F}_1^{v_1}(\boldsymbol{\xi}_{\kappa-1}, \boldsymbol{\zeta}_{\kappa-1}), \tag{48}$$

and then, with the newly computed $\boldsymbol{\xi}_\kappa$, do

$$\boldsymbol{\zeta}_\kappa = \boldsymbol{F}_2^{v_2}(\boldsymbol{\zeta}_{\kappa-1}, \boldsymbol{\xi}_\kappa). \tag{49}$$

This uses the new information as soon as it is available, and in the context of domain decomposition it would be called a multiplicative or serial Schwarz procedure [41]. The starting values $\boldsymbol{\xi}_0$ and $\boldsymbol{\zeta}_0$ are again given by the extrapolation mapping Eq. (14). This is the most commonly used procedure [30], and one iteration here is similar to one step in Eqs. (11) and (13) in Section 3.1.1.

Naturally, for both Eqs. (46), (47) and Eqs. (48), (49) one has to ask the question whether the iterative process converges. In this situation it is useful to note the following result [32,40,25,26].

Theorem 1. *In case the coupling equation is included in the first subsystem, let*

$$\alpha = \max_{t\in[0,T]} \|(D_{y_2}\boldsymbol{g}_2)^{-1} D_z \boldsymbol{g}_2 (D_{y_1}\boldsymbol{h}(D_{y_1}\boldsymbol{g}_1)^{-1} D_z \boldsymbol{g}_1)^{-1} D_{y_2}\boldsymbol{h}\|, \tag{50}$$

and in case the coupling is included in the second subsystem, let

$$\alpha = \max_{t\in[0,T]} \|(D_{y_2}\boldsymbol{h}(D_{y_2}\boldsymbol{g}_2)^{-1} D_z \boldsymbol{g}_2)^{-1} (D_{y_1}\boldsymbol{h}(D_{y_1}\boldsymbol{g}_1)^{-1} D_z \boldsymbol{g}_1)\|, \tag{51}$$

and let L be the Lipschitz constant of the extrapolation Eq. (14). Assume that $\alpha < 1$, and that at least κ iterations of the block-Gauss–Seidel scheme Eqs. (48) and (49) are performed, such that $L\alpha^\kappa < 1$, and that Δt is small enough. Then the global block-Gauss–Seidel method converges, and the global error in the n*th time step*

$$\delta^{(n)} = \|\boldsymbol{x}^{(n)} - \boldsymbol{x}(t_n)\| + \|\boldsymbol{y}^{(n)} - \boldsymbol{y}(t_n)\| + \|\boldsymbol{z}^{(n)} - \boldsymbol{z}(t_n)\|$$

is bounded by

$$\delta^{(n)} \leqslant C(\mu^{\max\{0,\kappa-2\}}(\delta^{(n)}\boldsymbol{x}) + \mu^{\kappa-1}(\delta^{(n)}\boldsymbol{y})) + \varepsilon_1^{(n)} + \varepsilon_2^{(n)}. \tag{52}$$

Here $\varepsilon_1^{(n)}$ and $\varepsilon_2^{(n)}$ are the errors incurred by the single system integrators given in Eqs. (23)–(25), $\delta^{(n)}\boldsymbol{y}$ and $\delta^{(n)}\boldsymbol{z}$ are the extrapolation errors, and $\mu = \alpha + \mathrm{O}(\Delta t)$.

If $\alpha < 1$ and Δt is such that $\mu < 1$, the iteration will converge. The contraction constant α is crucial, and convergence depends strongly on the ordering of the subsystems in the block-Gauss–Seidel solution strategy.

If enough iterations are performed, essentially only the error components from the single system integrators remain. If $\varepsilon_1^{(n)} = \mathrm{O}((\Delta t)^p)$ and $\varepsilon_2^{(n)} = \mathrm{O}((\Delta t)^q)$ are the convergence orders for the single system integrators, we obtain $\delta^{(n)} = \mathrm{O}((\Delta t)^{\min(p,q)})$, in contrast to the staggering scheme where we only have $\mathrm{O}(\Delta t)$ in the simplest case. This may be the distinctive advantage of full coupling, even if the staggering scheme is stable.

In the following a summary is given for FSI as how the solution strategy, the grouping of the subsystems and algebraic constraint equations, and the order in the Gauss–Seidel process determine the contraction constant α. The computations leading to this are a bit tedious but straightforward, and are omitted [25,26].

- Assume the subsystems and ordering are first fluid and coupling conditions, and second the structure. Then the contraction constant in Eq. (50) is

$$\alpha = \|\boldsymbol{M}_{\mathrm{s}}^{-1}\boldsymbol{T}_{\mathrm{s}}^{\mathrm{T}}(\boldsymbol{T}_{\mathrm{f}}\widehat{\boldsymbol{M}}^{-1}\boldsymbol{T}_{\mathrm{f}}^{\mathrm{T}})^{-1}\boldsymbol{T}_{\mathrm{s}}\|, \tag{53}$$

where $\widehat{\boldsymbol{M}}^{-1} = \boldsymbol{M}_{\mathrm{f}}^{-1}(\boldsymbol{M}_{\mathrm{f}} - \boldsymbol{B}_{\mathrm{f}}\widetilde{\boldsymbol{M}}_{\mathrm{p}}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}})\boldsymbol{M}_{\mathrm{f}}^{-1}$ is a Schur complement, and $\widetilde{\boldsymbol{M}}_{\mathrm{p}} = (\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{M}_{\mathrm{f}}^{-1}\boldsymbol{B}_{\mathrm{f}})^{-1}$ is a transformed mass matrix acting on pressures.
- Now assume the subsystems and ordering are first structure and coupling conditions, and second the fluid. The contraction constant in Eq. (50) is

$$\alpha = \|\widehat{\boldsymbol{M}}^{-1}\boldsymbol{T}_{\mathrm{f}}^{\mathrm{T}}(\boldsymbol{T}_{\mathrm{s}}\boldsymbol{M}_{\mathrm{s}}^{-1}\boldsymbol{T}_{\mathrm{s}}^{\mathrm{T}})^{-1}\boldsymbol{T}_{\mathrm{f}}\|, \tag{54}$$

which looks "inverse" to the previous relation Eq. (53).
- Assume the ordering and subsystems as first the fluid, and second the structure and coupling conditions. Then the contraction constant Eq. (51) is

$$\alpha = \|(\boldsymbol{T}_{\mathrm{s}}\boldsymbol{M}_{\mathrm{s}}^{-1}\boldsymbol{T}_{\mathrm{s}}^{\mathrm{T}})^{-1}(\boldsymbol{T}_{\mathrm{f}}\widehat{\boldsymbol{M}}^{-1}\boldsymbol{T}_{\mathrm{f}}^{\mathrm{T}})\|. \tag{55}$$

- Finally, assume the following subsystems and ordering: First the structure, and second the fluid and the coupling conditions. Here the contraction constant Eq. (51) is

$$\alpha = \|(\boldsymbol{T}_{\mathrm{f}}\widehat{\boldsymbol{M}}^{-1}\boldsymbol{T}_{\mathrm{f}}^{\mathrm{T}})^{-1}(\boldsymbol{T}_{\mathrm{s}}\boldsymbol{M}_{\mathrm{s}}^{-1}\boldsymbol{T}_{\mathrm{s}}^{\mathrm{T}})\|, \tag{56}$$

which is the norm of the inverse of the matrix in Eq. (55).

These relations show how the block-Gauss–Seidel process depends on the ordering and grouping of the equations. As $\widehat{\boldsymbol{M}}$ scales with the fluid density ϱ_{f}, and $\boldsymbol{M}_{\mathrm{s}}$ scales with the structural density ϱ_{s}, the contraction constant α is essentially determined by the density ratio $\varrho_{\mathrm{f}}/\varrho_{\mathrm{s}}$ in Eqs. (53) and (56), respectively, the inverse $\varrho_{\mathrm{s}}/\varrho_{\mathrm{f}}$ in Eqs. (54) and (55), something which has often been observed by practitioners.

In our view this strong dependence on ordering and grouping in the Gauss–Seidel process calls for strong or tight coupling methods which will converge unconditionally provided the time step is small enough; this would roughly be a situation similar for the single system implicit integrator. With block-Gauss–Seidel methods there is no easy way of achieving this, although there are possibilities of preconditioning [32].

5.3. Inexact block-Newton

Such an algorithm can be sought in the group of Newton-like methods, the full Newton–Raphson method is considered here. It is also in this context the more challenging, as it requires the solution of a global system of linear equations in each iteration, and especially as in the partitioned approach one has no recourse to the cross-coupling parts of the system matrix. With the Newton algorithm one has a robust method, where the convergence behaviour does not depend on

- in which subsystem the coupling is included,
- and in which order the subsystems are solved.

Usually one would start from Eqs. (23)–(25), the "equilibrium" equations; however it has been noted [37] that the solutions of Eqs. (23)–(25) are solutions of the fixed point problem Eqs. (44) and (45) and vice versa, but the latter are numerically much better conditioned as they implicitly already have the effect of the single system solvers in them. Hence, performing the Newton–Raphson iteration on the equations $\xi = \boldsymbol{F}_1(\xi,\zeta)$, $\zeta = \boldsymbol{F}_2(\zeta,\xi)$ corresponds to the iteration Eq. (57), with $\Delta\xi_\kappa := \xi_{\kappa+1} - \xi_\kappa$ and $\Delta\zeta_\kappa := \zeta_{\kappa+1} - \zeta_\kappa$, and the iteration counter κ:

$$\begin{bmatrix} \boldsymbol{I} - D_\xi \boldsymbol{F}_1 & -D_\zeta \boldsymbol{F}_1 \\ -D_\xi \boldsymbol{F}_2 & \boldsymbol{I} - D_\zeta \boldsymbol{F}_2 \end{bmatrix} \begin{bmatrix} \Delta \xi_\kappa \\ \Delta \zeta_\kappa \end{bmatrix} = - \begin{bmatrix} \xi_\kappa - \boldsymbol{F}_1(\xi_\kappa, \zeta_\kappa) \\ \zeta_\kappa - \boldsymbol{F}_2(\zeta_\kappa, \xi_\kappa) \end{bmatrix}. \tag{57}$$

In the proposed approach one only wants to use the existing solvers, i.e. the iteration mappings $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ from Eqs. (44) and (45). In particular, there is often no direct access to the cross terms in the global system matrix in Eq. (57). However if the system equation (57) is solved by a Krylov-type method [39,25,35,36], all one needs is a way to compute the product of the Jacobian matrix in Eq. (57) with an arbitrary vector, at least approximately. This can be viewed as a directional derivative. Briefly this can be described as follows: An approximation of the application of the diagonal blocks in Eq. (57) can be achieved by applying the subsystem solvers Eqs. (44) and (45). Application of the off-diagonal blocks to a vector can be approximated simply by finite differences, again using the subsystem solvers. A detailed description of one implementation of the block-Newton method can be found in [25,26].

5.4. Quasi-Newton

Another approach to realise a strong coupling is to apply a quasi-Newton solver like the Broyden–Fletcher–Goldfarb–Shanno (BFGS) [68–71,66,67] method directly to the equilibrium Eqs. (23)–(25). By having a preconditioner respectively starting matrix $\boldsymbol{H}_0$ corresponding (see Eqs. (58) and (59)) to the block-Jacobi matrix, this is almost equivalent to using the BFGS-method on the iteration fixed point Eqs. (44) and (45) with the identity as preconditioner respectively starting matrix. This method has an iteration scheme very similar to the Newton–Raphson-method, but instead uses approximations of the Jacobian. These approximations are defined by the recursion:

$$\text{starting matrix:} \quad \boldsymbol{H}_0 \tag{58}$$

$$\boldsymbol{H}_{k+1} := \boldsymbol{H}_k + \boldsymbol{a}_k \boldsymbol{a}_k^{\mathrm{T}} + \boldsymbol{b}_k \boldsymbol{b}_k^{\mathrm{T}} \quad (k \geqslant 0), \tag{59}$$

where the $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$ are vectors computed inexpensively from changes in the approximate solution and the equation residual in the iteration process. If the operation $\boldsymbol{H}_0^{-1}$ is available, then, by the Sherman–Morrison–Woodbury formula, the operation of $\boldsymbol{H}_k^{-1}$ on a vector is also easily computed, cf. [68–71,66,67], and is given by a formula analogous to Eq. (59). For large-scale computations it is also important not compute the updated matrix explicitly (it would be a full matrix), but to keep the factors of the dyadic or tensor products in Eq. (59) as individual vectors, as the explicit matrix is never needed, but only its action on a vector.

The main advantage of the BFGS algorithm is, that it needs only the residual and no derivatives. Therefore one has only to implement the total residual function, resp. one call for the subsystem solvers. Furthermore this method can use the subsystem solver or—if present—the preconditioners of the simulations in order to improve the convergence of the nonlinear iteration.

A potential disadvantage is that the vectors $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$ need to be stored. However, this can be alleviated by restarting [66], i.e. periodically "forgetting" all update vectors, or just replacing the old ones, once a certain number of updates has been reached.

5.5. Parallelisation

The parallelisation aspects of the solvers correspond in many ways to the well-known analogous properties of the solution of linear systems. The computation of the residuals can in all cases naturally be performed in parallel.

The block-Jacobi iteration, cf. Section 5.1, can be naturally parallelised, it corresponds to an additive Schwarz-method. Each block of the iteration can be performed independently of the others, and may itself be a—lower level—parallel code.

As is well known, the block-Gauss–Seidel iteration, cf. Section 5.2, uses the solvers $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ in a serial manner, it corresponds to a multiplicative Schwarz-method. Each component solver by itself of course may be a parallel code.

The inexact block-Newton, cf. Section 5.3, can also use the individual solvers in parallel, see [25].

The preconditioner in the BFGS iteration, cf. Section 5.4, which corresponds to one block-Jacobi iteration, is the most time consuming part, and can therefore be performed in parallel. The sum of the action of the tensor or dyadic products in Eq. (59) can again naturally be parallelised.

The used techniques for parallelisation are described in Section 7 in some detail. Comparisons of the convergence and computational times using a simple example will be given in Section 8.1.

6. Simulation interface and its requirements

In this paper a partitioned (i.e. non-monolithic) approach to solve coupled systems is given. Hence we need not only an algebraic formulation, but also a software realisation of the coupling of existing simulation codes.

One purely synchronous approach is to insert communication points into the codes, where the needed data transfer between the simulations is to be performed. In this case experts for both simulations components are needed in order to find the correct lines in the source code of the programs.

We have chosen an asynchronous functional approach. This suggests a standardisation of simulation codes in form of an interface definition, which has to be implemented by the different simulation codes. It is designed for the case that an external solver like the block-Newton method or a Gauss–Seidel-iteration can be applied to solve the coupled system of equations. Any access of such a solver to the individual simulations is done via methods of this interface, so that the simulations become exchangeable, or in other words, arbitrary structural or fluid simulations implementing this interface can be coupled. The individual subsystem codes appear globally as software objects in the sense of an object-oriented approach, implementing certain methods.

6.1. The simulation interface

This interface is formulated in the interface definition language of the component library CTL briefly described in Section 7. Therefore it can be used and implemented by C, Fortran, or C^{++} codes, and enables remote method invocation needed for a distributed simulation.

```
#define CTL_Class simuRI
#include CTL_ClassBegin
// init simulation with “filename” as starter file
# define CTL_Method1 void, init,\
    (const string / * filename * /), 1
// get state variables into x
# define CTL_Method2 void, getstate,\
    (array⟨real8⟩/ * x * /) const, 1
// set state variables to x
# define CTL_Method3 void, setstate,\
    (const array⟨real8⟩/ * x * /), 1
// set load of system
# define CTL_Method4 void, setload,\
    (const array⟨real8⟩/ * load * /), 1
```

```
// get coupling indices i and values y
// (bound.cond.or load or ...)
# define CTL_Method5 void, getcoupling,\
    (array⟨int4⟩/ * i * /, array⟨real8⟩/ * y * /) const, 2
// set coupling values y (bound.cond.or load or ...)
# define CTL_Method6 void, setcoupling,\
    (const array⟨real8⟩/ * y * /), 1
// compute res.at given state and write it into r
# define CTL_Method7 void, residual,\
    (array⟨real8⟩/ * r * /) const, 1
// solve with given load + coupling values
// with at least accuracy
// set new state and write it into x
# define CTL_Method8 void, solve,\
    (const real8 / * accuracy * /, array⟨real8⟩/ * x * /), 2
// compute a preconditioning step on r
// write result into pr
# define CTL_Method9 void, precond,\
    (const array⟨real8⟩/ * r * /,\
    array⟨real8⟩/ * pr * /) const, 2
// compute the directional derivative in x0
// in direction dx and write it into dr
# define CTL_Method10 void, dirderivative,\
    (const array⟨real8⟩/ * x0 * /,\
    const array⟨real8⟩/ * dx * /,\
    array⟨real8⟩/ * dr * /) const, 3
    ...
#include CTL_ClassEnd
```

The CTL uses the C-preprocssor for the generation of the code which handles all needed communications. Therefore the syntax of this interface uses `#define` and `#include` directives. Here a new class named `simuRI` is defined, which has the ten methods declared between line `#include CTL_ClassBegin` and `#include CTL_ClassEnd`.

The C^{++} representation of the defined class would look like

```
class simuRI
{
    void init(const string &filename);
    void getstate(vector⟨double⟩&x) const;
    ...
}
```

and the *Fortran* representation like

```
subroutine simu_init(size, filename)
integer * 8 size
character filename(size)
```

```
subroutine simu_getstate(size, x)
integer * 8 size
real * 8 x(size)
    ...
      .
```

Requirements to a simulation code are that its functionality can be split into the functions listed in the interface above, and—due to the functional approach—that a method invocation has no more side-effects than those given in the comments to this method. This implementation assumes that the simulation components know about the coupling, i.e. they have the information in which manner which degrees of freedom are coupled. Therefore the coupling solver itself can be formulated in purely algebraic terms.

6.2. Requirements of the solver

Beside the block-Newton method we implemented—formulated in the interface above—the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) and an inexact Newton method as solver for the coupled system. These solvers have different requirements to the simulations.

- the BFGS-algorithm only needs the residual (via `residual`) and optionally preconditioning (via `precond`),
- Jacobi- and Gauss–Seidel iteration access the internal solver of the simulation (via `solve`),
- the block-Newton method needs the residual and for the linear iterative solver directional derivatives of the residual (via `dirderivative`).
- the inexact Newton method the residual and optionally preconditioning and for the linear iterative solver directional derivatives of the (preconditioned) residual (via `dirderivative`).

If `precond` is not implemented by a simulation code no preconditioning is performed, if the implementation of `dirderivative` is missing the needed directional derivatives are approximated by finite differences.

7. Software architecture

The middle-ware used to implement the coupling is the *Component Template Library* (CTL), a recent development of our institute. This library serves as an easy to use programming environment for distributed applications in an abstract manner. It allows to define a new component, e.g. a simulation instance, using the functionality of an existing library without need of change nor recompilation of the library. Hence, if there is a library version of a simulation code, there is the possibility to reuse this code without having to know every detail of it.

Fig. 1 shows the dependency graph of the partitioned simulation. The building dependencies on the bottom level are given by compilation and linkage, the upper dependencies by file inclusion. There are three main parts, the simulations `simuA` and `simuB`, and the solver between them. All dependencies of these parts are centrally located in the interface header `simuInterface.h`. The header `simuA.h` and `simuB.h` are used by the sources `simuALink.cc` and `simuBLink.cc` in order to link the functionality of the simulations to the interface.

At run-time the solver component `coupleSolver.exe` is linked to the simulation components either by remote method invocation using the `simuA.exe/simuB.exe` or by dynamic linkage using the `simuA.so/simuB.so`. The latter variant should be chosen if computation in parallel is not possible, as in the Gauss–Seidel iteration.

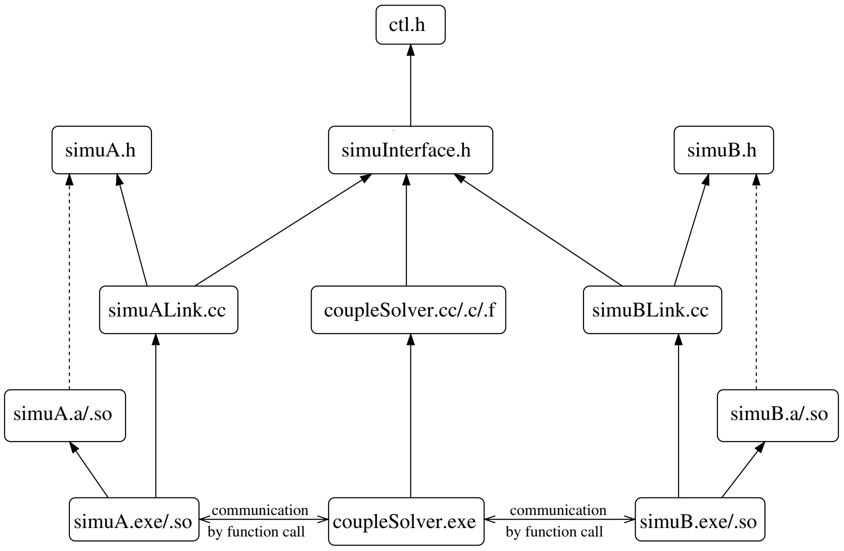


Fig. 1. Source dependencies of the coupling architecture.

The inexact Newton and the BFGS algorithm use a slightly different architecture. In this case there are two solver instances each linked to a simulation `simuA.so/simuB.so`. The solver instances communicate with each other in order to compute global residuals and scalar products.

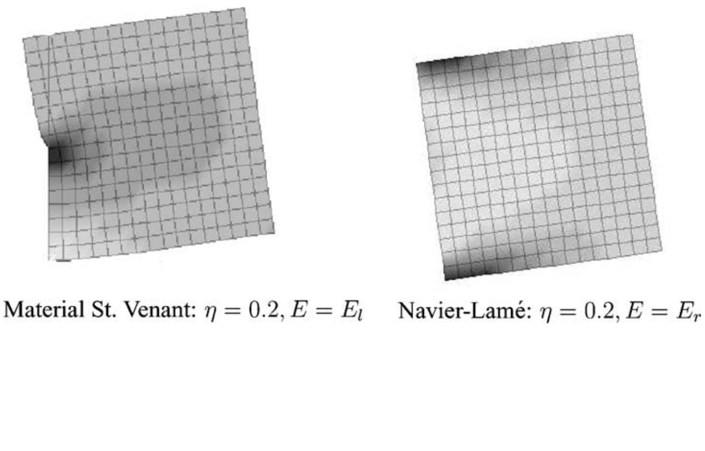
This architectures give an easy exchangeability of the simulation components without change nor recompilation of the solver as well as the possibility for distributed simulations. Using the functional approach and the described software architecture, the implementation of a coupling solver is very similar to serial programming. The complexity of message passing libraries like MPI or PVM is completely hidden.

8. Numerical examples

8.1. Simple example of a non-linear coupling

The simple structural example points out the limits of the coupling Jacobi- and Gauss–Seidel iterations. This system might arise in the simulation of a gasket consisting of plastic ring as the left and a metallic ring as the right part.

The lower left half of the left part is fixed by a homogeneous Dirichlet condition, whereas the right side of the right part is uniformly loaded.


Material St. Venant: $\eta = 0.2, E = E_l$
Navier-Lamé: $\eta = 0.2, E = E_r$

Due to the boundary conditions the left partial structure is much more distorted and was simulated with a geometrically nonlinear formulation. For the right part a linear formulation was chosen.

	$E_l = 10^5$, $E_r = 5 \times 10^4$		$E_l = 10^5$, $E_r = 10^6$		$E_l = 5 \times 10^4$, $E_r = 10^6$	
	iter	cpu [t]	iter	cpu [t]	iter	cpu [t]
Jacobi	18	10.9	∞	–	∞	–
Gauss–Seidel	14	7.9	28	16.0	∞	–
BFGS	56	6.8	44	4.8	110	13.4
Newton	3	13.5	3	13.7	12	52.0

In the Dirichlet–Neumann coupling of the partial structures the left part must be coupled by Neumann conditions and the right one by Dirichlet conditions. Otherwise the partial right system is not regular. For different elasticities we tried to solve the coupled system with the Jacobi-, Gauss–Seidel-, quasi-Newton, and the inexact Newton iteration. In the case $E_l = 5 \times 10^4$, $E_r = 10^6$ weak coupling does not work. This can be explained by the contractivity constant α which is in this coupling proportional to E_r/E_l. The reason for the increased iteration numbers of the other solver is the stronger non-linearity in the left part due to the smaller modulus. For all applied solvers the convergence behaviour can be improved by preconditioning of the global system [32].

8.2. *Rigid block with elastic appendage in incompressible viscous flow*

The DAE-formulation and the block-Newton method will be demonstrated [25] on a small two-dimensional example, introduced in [48] and already published in [26,27].

It is a rigid block with a thin elastic appendage, placed in an incompressible viscous flow coming from the left. The bluff shape of the square block causes separation and vortices, which are transported along the elastic appendage. Although the setup is symmetric, the vortices develop unsymmetrically, alternating from top and bottom. The pressure variations in the vortices interact with the elastic appendage, which in turn

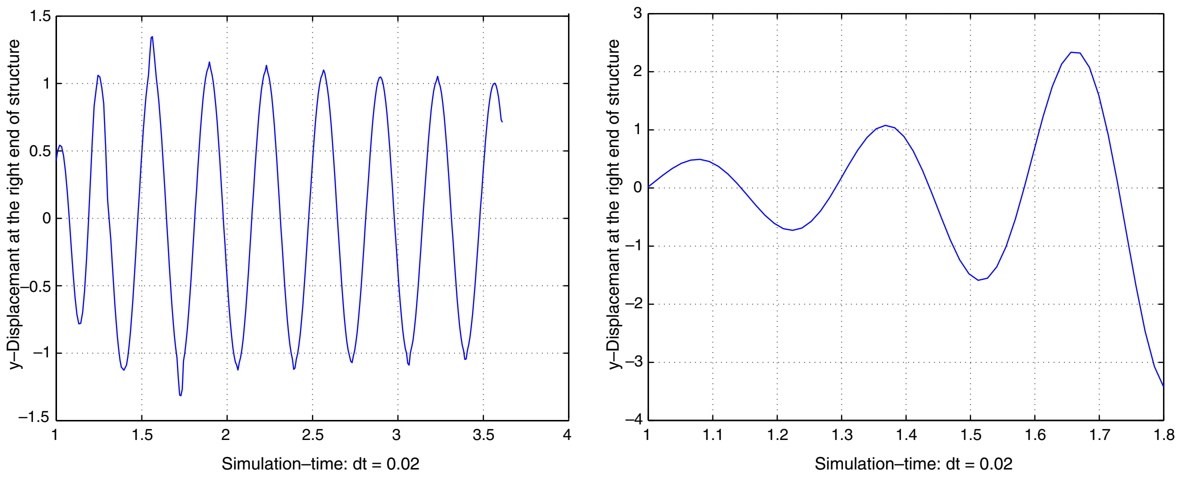


Fig. 2. Vortex shedding and pressure field.

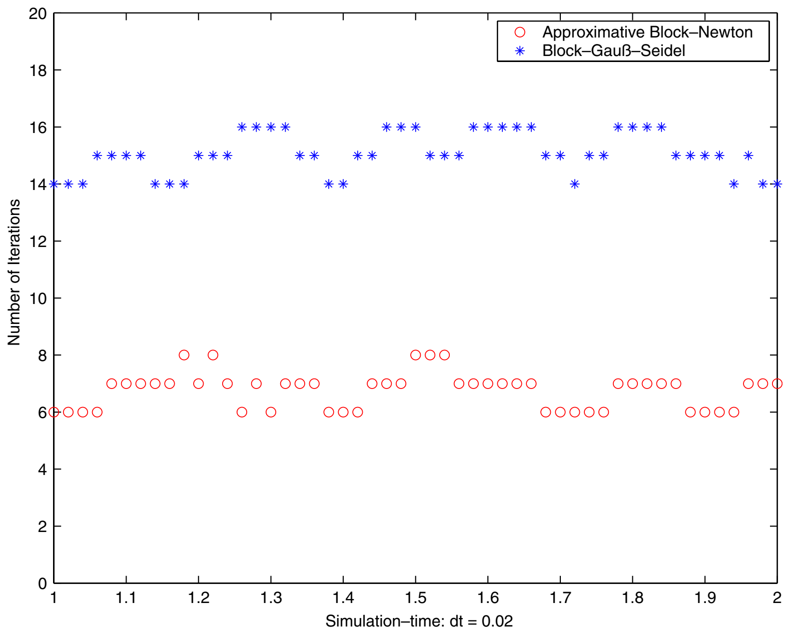


Fig. 3. Iteration count for block-Gauss–Seidel and block-Newton.

starts oscillating. This situation is shown in Fig. 2, together with the pressure field in the flow. More of the numerical behaviour of the coupling algorithm is shown in [26,27].

In another test, we compare the block-Gauss–Seidel method as described in Section 5.2 with the block-Newton method from Section 5.3. For the same configuration as before, we show the number of iterations of either method in each time step in Fig. 3. The superior convergence characteristic of the block-Newton method is obvious.

9. Conclusion

For coupled but *partitioned* problems, we have proposed a *strong* or *tight* coupling method, which achieves the same results as a monolithical approach. We have formulated this coupling problem as a differential algebraic equation (DAE) following [40,32]. As usually appropriate to the numerical treatment of DAEs, we consider globally implicit methods.

The global system to be solved in each time step is treated with the use of either the solvers or the residual function of the individual subsystems. The non-linear block-Jacobi and non-linear block-Gauss–Seidel methods come very naturally, but they are sub-optimal and not robust. We have discussed the problems which arise with this approach. We propose to solve the global implicit equations with a Newton-like method, following [37–39,25,26,36]. The linear system which arises in each iteration is solved with a Krylov iterative method.

We have demonstrated the methods on a fluid–structure interaction problem, first fitting it into the general framework, and then applying the proposed procedures. Needless to say that the general ideas about formulating and solving coupling problems are not specific to fluid–structure interaction and can be used in other circumstances as well.

The partitioned approach offers modularity both in the formulation and modelling of physical phenomena, and—a point of paramount practical importance—especially in the implementation of software [62].

This allows the use of existing software packages for the subsystems, and this was how the computations presented here were done. The structural solver was *FEAP* [64,11], and the flow solver used was *FEAT-FLOW* [65,72], which could be taken as building blocks and only needed a small interface to be able to perform the coupling [25].

The overall software architecture is based on an interface definition for simulations together with a high level programming environment designed to do the clerical work of managing the communication between different software packets. The various solvers have different requirements as to which parts of this simulation-interface has to be implemented by the simulation codes. With this kind of approach, the effort and special knowledge and techniques which have gone into the design and implementation of software for specific application areas can be re-used, and the coupled simulation can benefit from this, and things do not have to be designed from scratch again.

In closing, we would like to emphasise the fact that it is possible to have the above mentioned advantages of the partitioned approach, and at the same time the superior robustness and convergence characteristics of a monolithical approach—in our view this is the best of both worlds.

References

[1] H. Morand, R. Ohayon, Fluid–structure Interaction, John Wiley & Sons, Chichester, 1995.
[2] T. Belytschko, R. Mullen, Two-dimensional fluid–structure impact computations with regularization, Comput. Methods Appl. Mech. Engrg. 27 (1981) 139–154.
[3] J.M. Kennedy, T. Belytschko, A survey of computational methods for fluid–structure analysis of reactor safety, Nucl. Engrg. Des. 69 (1982) 379–398.
[4] J. Donea, S. Giuliani, J.P. Halleux, An arbitrary Lagrangian–Eulerian finite element method for transient fluid–structure interaction, Comput. Methods Appl. Mech. Engrg. 33 (1982) 689–723.
[5] K.-J. Bathe, H. Zhang, M.H. Wang, Finite element analysis of incompressible and compressible fluid flows with free surfaces and structural interactions, Comput. Struct. 56 (1995) 193–213.
[6] S. Piperno, C. Farhat, B. Larrouturou, Partitioned procedures for the transient solution of coupled aeroelastic problems, Comput. Methods Appl. Mech. Engrg. 124 (1995) 79–112.
[7] J. Mouro, P. Le Tallec, Fluid structure interaction with large structural displacements, Comput. Methods Appl. Mech. Engrg. 190 (2001) 3039–3067.
[8] C. Farhat, Parallel and distributed solution of coupled nonlinear dynamic aeroelastic response, in: M. Papadrakakis (Ed.), Parallel Solution Methods in Computational Mechanics, John Wiley & Sons, Chichester, 1997.
[9] S. Rifai, Z. Johan, W.-P. Wang, J.-P. Grisval, T.J.R. Hughes, M. Ferencz, Multiphysics simulation of flow induced vibrations and aeroelasticity on parallel computing platforms, Comput. Methods Appl. Mech. Engrg. 174 (1999) 393–417.
[10] K.-J. Bathe, H. Zhang, S.H. Ji, Finite element analysis of fluid flows fully coupled with structural interactions, Comput. Struct. 72 (1999) 1–16.
[11] O.C. Zienkiewicz, R.L. TaylorThe Finite Element Method, vol. 1–3, Butterworth Heinemann, London, 2001.
[12] J. Argyris, I.S. Doltsinis, P. Pimenta, H. Wüstenberg, Thermomechanical response of solids at high strains—natural approach, Comput. Methods Appl. Mech. Engrg. 32 (1982) 3–57.
[13] O.C. Zienkiewicz, D.K. Paul, A.H.C. Chan, Unconditionally stable staggered solution procedure for soil–pore fluid interaction problems, Int. J. Numer. Methods Engrg. 26 (1988) 1039–1055.
[14] K.C. Park, C.A. Felippa, Partitioned analysis of coupled systems, in: T. Belytschko, T.J.R. Hughes (Eds.), Computational Methods in Transient Analysis, North-Holland, Amsterdam, 1983.
[15] K.C. Park, C.A. Felippa, Recent developments in coupled field analysis methods, in: R.W. Lewis, P. Bettes, E. Hinton (Eds.), Numerical Methods in Coupled Systems, John Wiley & Sons, Chichester, 1984.
[16] O.C. Zienkiewicz, Coupled problems and their numerical solution, in: R.W. Lewis, P. Bettes, E. Hinton (Eds.), Numerical Methods in Coupled Systems, John Wiley & Sons, Chichester, 1984.
[17] O.C. Zienkiewicz, A.H.C. Chan, Coupled problems and their numerical solution, in: I.S. Doltsinis (Ed.), Advances in Computational Nonlinear Mechanics, Springer-Verlag, Berlin, 1989.
[18] F.J. Blom, A monolithical fluid–structure interaction algorithm applied to the piston problem, Comput. Methods Appl. Mech. Engrg. 167 (1998) 369–391.
[19] S. Rugonyi, K.-J. Bathe, On the analysis of fully-coupled fluid flows with structural interactions—a coupling and condensation procedure, Int. J. Comput. Civil Struct. Engrg. 1 (2000) 29–41.

[20] C.A. Felippa, K.C. Park, Staggered transient analysis procedures for coupled mechanical systems: formulation, Comput. Methods Appl. Mech. Engrg. 24 (1980) 61–111.
[21] K.C. Park, Partitioned transient analysis procedures for coupled-field problems: stability analysis, J. Appl. Mech. 47 (1980) 370–376.
[22] S. Piperno, Explicit/implicit fluid–structure staggered procedures with a structural predictor and fluid subcycling for 2D inviscid aeroelastic simulations, Int. J. Numer. Methods Fluids 25 (1997) 1207–1226.
[23] D.P. Mok, W.A. Wall, Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures, in: W.A. Wall, K.-U. Bletzinger, K. Schweizerhof (Eds.), Trends in Computational Structural Mechanics, CIMNE, Barcelona, 2001.
[24] J. Steindorf, H.G. Matthies, Efficient partitioned methods for the computation of fluid–structure interaction on parallel computers, in: B.H.V. Topping (Ed.), Proceedings of the Third Euro-conference on Parallel and Distributed Computing for Computational Mechanics, Civil-Comp Press, Edinburgh, 1999.
[25] J. Steindorf, Partitionierte Verfahren für Probleme der Fluid–Struktur Wechselwirkung, Doctoral thesis, Technische Universität Braunschweig, Brunswick, 2002.
[26] H.G. Matthies, J. Steindorf, Strong coupling methods, in: W.L. Wendland, M. Efendiev (Eds.), Analysis and Simulation of Multifield Problems, Springer-Verlag, Berlin, 2003.
[27] H.G. Matthies, J. Steindorf, Partitioned strong coupling algorithms for fluid–structure interaction, Informatik-Bericht 2002-4, Technische Universität Braunschweig, Brunswick, 2002.
[28] S. Piperno, C. Farhat, Design of efficient partitioned procedures for the transient solution of aeroelastic problems, Rev. Eur. Élements Finis 9 (6–7) (2000) 655–680.
[29] E. Lefrantois, G. Dhatt, D. Vandromme, Numerical study of the aeroelastic stability of an overexpanded rocket-nozzle, Rev. Eur. Élements Finis 9 (6–7) (2000) 727–762.
[30] P. Leyland, V. Carstens, F. Blom, T. Tefy, Fully coupled fluid–structure algorithms for aeroelasticity and forced vibration induced flutter: applications to compressor cascade, Rev. Eur Élements Finis 9 (6–7) (2000) 763–803.
[31] R. Codina, M. Cervera, Block-iterative algorithms for nonlinear coupled problems, in: M. Papadrakakis, G. Bugeda (Eds.), Advanced Computational Methods in Structural Mechanics, CIMNE, Barcelona, 1996.
[32] M. Arnold, M. Günther, Preconditioned dynamic iteration for coupled differential–algebraic systems, BIT Numer. Math. 41 (2001) 1–25.
[33] H.G. Matthies, J. Steindorf, Efficient iteration schemes for non-linear fluid–structure interaction problems, in: B.H.V. Topping (Ed.), Computational Mechanics: Techniques and Developments, Civil-Comp Press, Edinburgh, 2000.
[34] H.G. Matthies, J. Steindorf, How to make weak couplings strong, in: K.-J. Bathe (Ed.), Computational Fluid and Solid Mechanics, Elsevier, Amsterdam, 2001.
[35] H.G. Matthies, J. Steindorf, Fully coupled fluid–structure interaction using weak coupling, Proc. Appl. Math. Mech. 1 (1) (2002) 37–38.
[36] H.G. Matthies, J. Steindorf, Partitioned strong coupling algorithms for fluid–structure interaction, Comput. Struct. 81 (2003) 1277–1286.
[37] S. Artlich, W. Mackens, Newton-coupling of fixed point iterations, in: W. Hackbusch, G. Wittum (Eds.), Numerical Treatment of Coupled Systems, Vieweg, Brunswick, 1995.
[38] W. Mackens, J. Menck, H. Voss, Numerical system synthesis: concepts for coupling subsystem solvers, Technical Report, Technische Universität Hamburg-Harburg, 1998.
[39] W. Mackens, J. Menck, H. Voss, Numerical coupling of subsystems, Zeitschr. Angew. Math. Mech. 79 (3) (1999) S871–S872.
[40] M. Arnold, Constraint partitioning in dynamic iteration methods, Zeitschr. Angew. Math. Mech. 81 (2001).
[41] B. Smith, P. Bjorstad, W. Gropp, Domain Decomposition, Cambridge University Press, Cambridge, 1996.
[42] K. Burrage, Parallel and Sequential Methods for Ordinary Differential Equations, Clarendon Press, Oxford, 1995.
[43] T.F. Chan, An approximate Newton method for coupled nonlinear systems, SIAM J. Numer. Anal. 22 (5) (1985) 904–913.
[44] U. Miekkala, O. Nevanlinna, An approximate Newton method for coupled nonlinear systems, SIAM J. Sci. Stat. Comput. 8 (1987) 459–482.
[45] C. Farhat, M. Lesoinne, Two efficient staggered algorithms for the serial and parallel solution of three-dimensional transient aeroelastic problems, Comput. Methods Appl. Mech. Engrg. 182 (2000) 499–515.
[46] C. Farhat, K.C. Park, Y. Dubois-Pelerin, An unconditionally stable staggered algorithm for transient finite element analysis of coupled thermoelastic problems, Comput. Methods Appl. Mech. Engrg. 85 (1991) 349–365.
[47] J. Donea, Arbitrary Lagrangian–Eulerian finite element methods, in: T. Belytschko, T.J.R. Hughes (Eds.), Computational Methods in Transient Analysis, North-Holland, Amsterdam, 1983.
[48] W.A. Wall, E. Ramm, Fluid–structure interaction based upon a stabilized (ALE) finite element method, in: Proceedings of the 4th World Congress on Computational Mechanics—new Trends and Applications, CIMNE, Barcelona, 1998.
[49] K.-J. Bathe, Finite Element Procedures, Prentice-Hall, Englewood Cliffs, NJ, 1996.
[50] J.H. Ferziger, M. Perić, Computational Methods for Fluid Dynamics, Springer-Verlag, Berlin, 1996.

[51] P.D. Thomas, C.K. Lombard, Geometric conservation law and its application to flow computations on moving grids, AIAA J. 17 (10) (1979) 1030–1037.

[52] G. Dubini, R. Pietrabissi, F.M. Montevecchi, Fluid–structure interaction in bio-fluid mechanics, Med. Engrg. Phys. 17 (2) (1995) 609–617.

[53] C. Grandmont, V. Guimet, Y. Maday, Numerical analysis of some decoupling techniques for the approximation of the unsteady fluid–structure interaction, in: Proceedings of the Second European Conference on Numerical Mathematics, Heidelberg, 1997.

[54] J.T. Batina, Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA J. 28 (8) (1990) 1381–1388.

[55] J.R. Cebral, R. Löhner, Conservative load projection and tracking for fluid–structure problems, AIAA J. 35 (4) (1997) 687–692.

[56] C. Farhat, M. Lesoinne, P. Le Tallec, Load and motion transfer algorithms for fluid–structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity, Comput. Methods Appl. Mech. Engrg. 157 (1998) 95–114.

[57] N. Maman, C. Farhat, Matching fluid and structure meshes for aeroelastic computations: a parallel approach, Comput. Struct. 54 (1995) 779–785.

[58] B. Wohlmuth, Discretization Methods and Iterative Solvers Based on Domain Decomposition, Lecture Notes in Computational Science and Engineering, vol. 17, Springer-Verlag, Berlin, 2001.

[59] P. Le Tallec, S. Mani, Conservation laws for fluid–structure interactions, Technical Report CEREMADE, Université de Paris Dauphine, 1999.

[60] T. Belytschko, R. Mullen, Stability of explicit–implicit mesh partitions in time integration, Int. J. Numer. Methods Engrg. 12 (1978) 1575–1586.

[61] C. Farhat, M. Lesoinne, N. Maman, Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation law and distributed solution, Int. J. Numer. Methods Fluids 21 (1995) 807–835.

[62] C. Farhat, M. Lesoinne, P. Stern, High performance solution of three-dimensional nonlinear elastic problems via parallel partitioned algorithms: methodology and preliminary results, Adv. Engrg. Software 28 (1997) 43–61.

[63] U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential–algebraic Equations, SIAM, Philadelphia, PA, 1998.

[64] R.L. Taylor, FEAP—A finite element analysis program, User Manual Version 6.3; Dept. of Civil and Environmental Engrg., University of California, Berkeley, CA, 1998.

[65] S. Turek, C. Becker, FEATFLOW User Manual, Release 1.2; Institut für Angewandte Mathematik, Universität Heidelberg, Heidelberg, 1999.

[66] H. Matthies, G. Strang, The solution of nonlinear finite element equations, Int. J. Numer. Methods Engrg. 14 (1979) 1613–1626.

[67] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstraint Optimization and Nonlinear Equations, SIAM, Philadelphia, PA, 1996.

[68] C.G. Broyden, The convergence of a class of double-rank minimization algorithms 2, the new algorithm, J. Inst. Math. Appl. 6 (1970) 222–231.

[69] R. Fletcher, A new approach to variable-metric algorithms, Comput. J. 13 (1970) 317–322.

[70] D. Goldfarb, A family of variable-metric algorithms derived by variational means, Math. Comput. 24 (1970) 23–26.

[71] D.F. Shanno, Conditioning of quasi-Newton methods for function minimization, Math. Comput. 24 (1970) 647–656.

[72] S. Turek, Efficient Solvers for Incompressible Flow Problems: an Algorithmic Approach in View of Computational Aspects, Lecture Notes in Computational Science and Engineering, vol. 6, Springer-Verlag, Berlin, 1999.