14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11–15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

COMPARISON OF DIFFERENT QUASI-NEWTON TECHNIQUES FOR COUPLING OF BLACK-BOX SOLVERS

NICOLAS DELAISSÉ[1], TOON DEMEESTER[1], DIETER FAUCONNIER[2,3] AND JORIS DEGROOTE[1,3]

[1] Department of Electromechanical, Systems and Metal Engineering, Ghent University
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
Nicolas.Delaisse@UGent.be

[2] Department of Electromechanical, Systems and Metal Engineering, Ghent University
Technologiepark Zwijnaarde 46, 9052 Zwijnaarde, Belgium

[3] Flanders Make @ UGent – Core Lab EEDT-MP

Key words: Quasi-Newton, Coupling Algorithm, Fluid-Structure Interaction

Abstract. Fluid-structure interaction (FSI) problems are frequently solved using partitioned simulation techniques with black-box solvers, reusing reliable and optimized codes. These problems can principally be reduced to solving a root-finding problem. In case of strong coupling, pure Gauss-Seidel iterations between the structure and flow solvers are unstable for lower modes. In these cases, quasi-Newton techniques are used, which construct an approximation of the Jacobian or its inverse by reusing information from previous iterations and time steps.

Four different quasi-Newton techniques are compared: *the interface quasi-Newton algorithm with an approximation for the inverse of the Jacobian from a least-squares model* (IQN-ILS), *the interface block quasi-Newton algorithm with approximate Jacobians from least-squares models* (IBQN-LS), *the interface quasi-Newton technique with multiple vector Jacobian* (IQN-MVJ) and *the multi-vector update quasi-Newton technique* (MVQN). These coupling algorithms are differentiated based on whether the approximation of the Jacobian is performed for the entire black-box system (IQN-ILS and IQN-MVJ) or for both individual solvers (IBQN-LS and MVQN). Moreover, a distinction is made between methods which perform the approximation with either least-squares models (IQN-ILS and IBQN-LS) or multi-vector techniques (IQN-MVJ and MVQN).

Their performance is compared by solving a 1D flexible tube case, using the in-house coupling software CoCoNuT. Both the memory usage and number of iterations between structure and flow solvers in each time step are examined. The techniques using a multi-vector approach require explicit matrix construction, so that memory requirements scale quadratically, whereas the least-squares techniques have a matrix-free implementation, resulting in linear scaling. In terms of convergence they are comparable.

1 INTRODUCTION

Multi-physics problems, such as fluid-structure interaction (FSI), are present in various engineering disciplines. Due to their high complexity, these problems are frequently solved numerically. Two general approaches exist to perform these numerical simulations. In the first one, a problem specific solver is written to find the solution in a monolithic way. This solver takes the interaction between the governing

equations into account directly by solving them simultaneously [1]. In the second approach, the calculation is done in a partitioned way. Then, the solvers are isolated and evaluated separately. Therefore, they may be existing software packages, with proven reliability and maturity. The partitioned procedure establishes a coupling between the two solvers, avoiding the need to develop a dedicated solver.

Within the partitioned approaches, both weakly and strongly coupled techniques exist. The former will evaluate the solver only once in each time step, whereas the latter will iterate between the solvers to obtain the desired result. With weakly coupled methods, also called explicit or loose coupling, the equilibrium between fluid and structure is not exactly satisfied. They are typically suited for aero-elastic simulations with light and compressible fluids [2, 3]. Problems with a higher ratio of added mass to mass of the structure require strongly coupled techniques, also called implicit coupling. These are used for problems with dense and incompressible fluids. The coupling iterations enforce the equilibrium at the fluid-structure interface, but are also more expensive as each solver is evaluated multiple times within each time step. However, the cost of subsequent iterations within each time step normally decreases as the change between the iterations diminishes.

This work focuses on cases with high added mass and incompressible fluids for which the strongly coupled, partitioned approach is adopted. In these cases, simply iterating between the solvers using Gauss-Seidel iterations is typically not sufficient, as is demonstrated by several stability-analyses [4, 5, 6]. Most often, Newton-Raphson iterations are performed to overcome this issue. However, as the solvers are often black-boxes, their Jacobians are not accessible and therefore need to be approximated. Hence, these techniques are called quasi-Newton methods. Besides the advantage of needing no modification of or access to the solvers, these techniques only require the knowledge of data at the interface.

Historically, the first quasi-Newton technique in partitioned FSI was the interface block quasi-Newton method (IBQN-LS), developed by Vierendeels et al. [7]. Initially, the term reduced order models (ROMs) was used to denote this method. However, due to confusion with proper orthogonal decomposition (POD) and Galerkin methods, this was later changed. Subsequently, Degroote et al. devised the interface quasi-Newton method with approximation for the inverse of the Jacobian from a least-squares model (IQN-ILS) [8]. Later, Bogaers et al. introduced the multi-vector update quasi-Newton technique (MVQN) to overcome the specification of the reuse parameter q in IQN-ILS [9]. The same paper proposed to apply the same technique projected on the interface displacement. However, this was only implemented later by Lindner et al. [10] and Blom et al. [11] and is called interface quasi-Newton multiple vector Jacobian (IQN-MVJ).

In this work, the above mentioned quasi-Newton techniques will be compared in terms of mathematical aspects by dividing them in several classes, after which their performance will be evaluated in terms of number of iterations per time step and memory requirements.

2 INTERFACE QUASI-NEWTON TECHNIQUES

A general FSI problem consists of a structural domain Ω_s and a fluid domain Ω_f, in which the respective governing equations are solved after discretization. The boundaries of both domains are denoted $\Gamma_s = \partial\Omega_s$ and $\Gamma_f = \partial\Omega_f$ respectively and their intersection is called the fluid-structure interface $\Gamma_i = \Gamma_f \cap \Gamma_s$. For the coupling algorithm only the variables on the common boundary are of interest. Because the flow and structure solver typically have a different mesh size at this interface, interpolation is usually required. As the focus in this paper is on the coupling, it will be assumed that the meshes on both sides of this interface coincide. The displacement is represented by $\boldsymbol{x} \in \mathbb{R}^{\mathrm{u}\times 1}$, containing the com-

ponents of the displacement vectors $\vec{u}$. The forces on the fluid-structure interface are given by $\boldsymbol{y} \in \mathbb{R}^{\mathrm{w}\times 1}$, containing the pressure p and the components of the traction vectors $\vec{t}$.

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} u_{1,1} & \dots & u_{1,d} & u_{2,1} & \dots & u_{2,d} & \dots \end{bmatrix}^{\mathrm{T}} \\ \boldsymbol{y} &= \begin{bmatrix} p_1 & t_{1,1} & \dots & t_{1,d} & p_2 & t_{2,1} & \dots & t_{2,d} & \dots \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{1}$$

The first subscript refers to the grid point and the second to the vector component, with d denoting the dimension of the vector.

In order to reach the solution, two type of equilibrium conditions on the fluid-structure interface have to be fulfilled. Firstly, the velocities must be equal (kinematic condition):

$$\vec{v} = \frac{\mathrm{D}\vec{u}}{\mathrm{D}t}, \tag{2}$$

for all point on the interface Γ_i, where $\vec{v}$ is the velocity in the fluid domain. Secondly, according to the third Newtonian law, the forces must be equal in size, but opposite in sign (dynamic condition):

$$\vec{\sigma}_f \cdot \vec{n}_f = -\vec{\sigma}_s \cdot \vec{n}_s, \tag{3}$$

for all points on the interface Γ_i, where $\vec{n}_{f,s}$ is the unit normal vector pointing outwards and $\vec{\sigma}_{f,s}$ the stress tensor. The FSI problem is fully closed by applying appropriate boundary conditions on the remainder of the boundaries $\Gamma_s \setminus \Gamma_i$ and $\Gamma_f \setminus \Gamma_i$.

Further, the structure and flow solvers are concisely written by $\boldsymbol{\mathcal{S}}(\boldsymbol{y})$ and $\boldsymbol{\mathcal{F}}(\boldsymbol{x})$, respectively. This notation hides the dependence on previous time steps and on the variables outside of the common boundary. The convention is adopted to denote the output of a solver with a tilde, resulting in

$$\begin{aligned} \tilde{\boldsymbol{y}}^{n+1,k+1} &= \boldsymbol{\mathcal{F}}(\boldsymbol{x}^{n+1,k+1}) \\ \tilde{\boldsymbol{x}}^{n+1,k+1} &= \boldsymbol{\mathcal{S}}(\boldsymbol{y}^{n+1,k+1}) \end{aligned} \tag{4}$$

for the calculation in time step $n+1$ and iteration $k+1$. The index n denoting the time step and the index k indicating the iteration within that time step will be omitted for brevity, whenever possible.

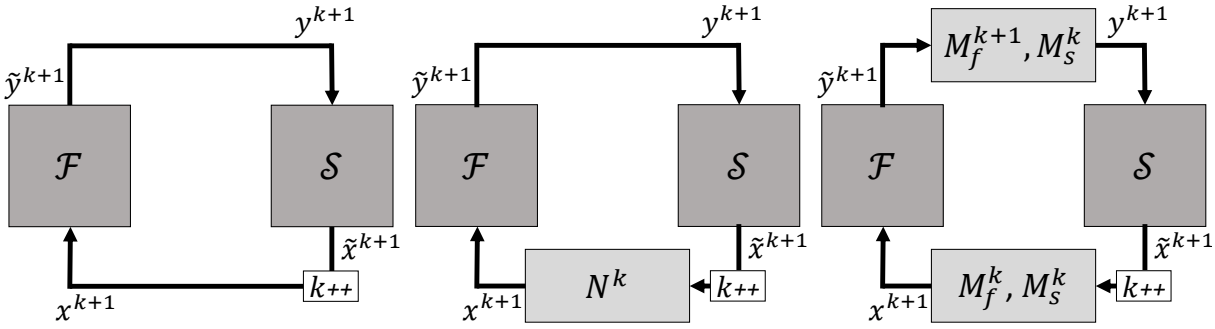
As such, solving the FSI problem boils down to finding $\boldsymbol{x}$ and $\boldsymbol{y}$ which satisfy the system

$$\begin{cases} \boldsymbol{\mathcal{F}}(\boldsymbol{x}) - \boldsymbol{y} = \boldsymbol{0} \\ \boldsymbol{\mathcal{S}}(\boldsymbol{y}) - \boldsymbol{x} = \boldsymbol{0} \end{cases} \tag{5}$$

within each time step n. Every time step is solved as follows. Firstly, interpolating the solution from previous time steps results in an initial guess. Thereafter, iterations are performed between the solvers until the inequalities

$$\begin{aligned} ||\boldsymbol{x}^{k+1} - \tilde{\boldsymbol{x}}^{k+1}||_2 &\leq \epsilon_x \\ ||\boldsymbol{y}^{k+1} - \tilde{\boldsymbol{y}}^{k+1}||_2 &\leq \epsilon_y \end{aligned} \tag{6}$$

are fulfilled. Here, ϵ_x and ϵ_y are convergence tolerances which can be absolute or relative to the value in the first iteration. The following part of this section deals with the different ways in which these iterations are performed.



(a) Gauss-Seidel scheme. **(b)** Residual formulation scheme. **(c)** Block iteration scheme.

Figure 1: Schematic representation of iteration schemes.

2.1 Three iteration schemes

Gauss-Seidel iterations

The simplest way to iterate between the structure and flow solver is to perform Gauss-Seidel iterations. In this scheme the output of one solver becomes the input of the next, until the change of $\boldsymbol{x}$ and $\boldsymbol{y}$ is no longer significant, i.e. Eq. (6) is fulfilled and convergence is reached. This iteration scheme is also called serial, because the solvers are solved consecutively. The parallel variant is called the Jacobi iteration procedure, in which the solvers are executed in parallel. For all methods discussed hereafter, a parallel twin exists. However, in this work the focus will solely be on serial iterations. For a comparison between both schemes refer to [10].

It is common, but rather arbitrary, to execute the flow solver first in each coupling iteration, followed by the structure solver. From that viewpoint, Eq. (5) may be written as

$$\tilde{\boldsymbol{x}}^{k+1} = \boldsymbol{\mathcal{S}} \circ \boldsymbol{\mathcal{F}}(\boldsymbol{x}^{k+1}), \tag{7}$$

where $\circ$ refers to function composition and is equivalent to $\boldsymbol{\mathcal{S}}(\boldsymbol{\mathcal{F}}(\boldsymbol{x}))$. As such, the problem is rewritten in function of the interface displacement only. A basic schematic is shown in Figure 1a. Furthermore, a residual operator $\boldsymbol{\mathcal{R}}$ is defined by

$$\boldsymbol{\mathcal{R}}(\boldsymbol{x}^{k+1}) = \boldsymbol{\mathcal{S}} \circ \boldsymbol{\mathcal{F}}(\boldsymbol{x}^{k+1}) - \boldsymbol{x}^{k+1}. \tag{8}$$

which may be rewritten using Eq. (7) as

$$\boldsymbol{\mathcal{R}}(\boldsymbol{x}^{k+1}) = \tilde{\boldsymbol{x}}^{k+1} - \boldsymbol{x}^{k+1} = \boldsymbol{r}^{k+1}, \tag{9}$$

and has an output $\boldsymbol{r}$, which will further be called the residual. This reduces the problem to solving

$$\boldsymbol{\mathcal{R}}(\boldsymbol{x}) = \boldsymbol{0}, \tag{10}$$

a set of non-linear equations for $\boldsymbol{x}$. Convergence is reached when $||\boldsymbol{r}||_2 \leq \epsilon_x$.

For FSI problems with high added mass and an incompressible fluid, Gauss-Seidel iterations are often unstable [4, 5, 6]. One way to address this problem is to introduce a scalar relaxation factor ω, which mixes new and old input.

$$\boldsymbol{x}^{k+1} = (1-\omega)\boldsymbol{x}^k + \omega\tilde{\boldsymbol{x}}^k = \boldsymbol{x}^k + \omega\boldsymbol{r}^k \tag{11}$$

A smaller relaxation factor increases the parameter range for which Gauss-Seidel iterations are stable, but penalizes the convergence speed. To avoid this cumbersome compromise, a dynamically varying relaxation factor can be used, as with Aitken relaxation [12]. In the latter technique, the changing value of ω^k is determined based o the previous value ω^{k-1} and the residual vectors $\boldsymbol{r}^k$ and $\boldsymbol{r}^{k-1}$.

Newton-Raphson iterations for $\boldsymbol{\mathcal{R}}$

As stated above, the FSI problem can reduced to solving Eq. (10) for $\boldsymbol{x}$. A typical method for solving such a set of non-linear equations, is to perform Newton-Raphson iterations. These make use of the inverse Jacobian of $\boldsymbol{\mathcal{R}}$, written symbolically as $\boldsymbol{\mathcal{R}}'^{-1} = \left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\boldsymbol{x}}\right)^{-1} = \left(\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{r}}\right)$, to obtain a better guess $\boldsymbol{x}^{k+1}$ for the next input as follows

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \boldsymbol{\mathcal{R}}'^{-1}(\boldsymbol{x}^k)\boldsymbol{r}^k, \tag{12}$$

This may also be regarded as a relation between differences

$$\Delta\boldsymbol{x}^k = \boldsymbol{\mathcal{R}}'^{-1}(\boldsymbol{x}^k)\Delta\boldsymbol{r}^k, \tag{13}$$

where $\Delta\boldsymbol{x}^k = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$ is the difference between the input of two subsequent iterations and $\Delta\boldsymbol{r}^k = \boldsymbol{0} - \boldsymbol{r}^k = -\boldsymbol{r}^k$ is the difference between the desired and the current residual. Likewise, $\Delta\tilde{\boldsymbol{x}}^k = \tilde{\boldsymbol{x}}^{k+1} - \tilde{\boldsymbol{x}}^k$ is the difference between the the structural solver output of two subsequent iterations.

A basic schematic is shown in Figure 1b. Note that the output of the flow solver is still passed on directly to the structural solver and hence the dynamic condition is exactly fulfilled in every iteration. In contrast to Gauss-Seidel iterations, the output of the structural solver is modified before it is transferred to the flow solver.

However, the Jacobian is not accessible in case of black-box solvers. Therefore, the inverse Jacobian $\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\boldsymbol{x}}\right)^{-1}$ is approximated and its estimation is denoted by $\boldsymbol{N}^k \in \mathbb{R}^{\mathrm{u}\times\mathrm{u}}$. Because this matrix needs to be full rank to serve its purpose in Newton-Raphson iterations, the approximation procedure discussed later will be not be applied to $\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\boldsymbol{x}}\right)^{-1}$, but to $\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\tilde{\boldsymbol{x}}}\right)^{-1}$ instead. Here $\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\tilde{\boldsymbol{x}}}\right)^{-1}$ symbolizes the inverse Jacobian of $\boldsymbol{\mathcal{R}}$ with respect to $\tilde{\boldsymbol{x}}$. Both inverse Jacobians are linked by the following equation

$$\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\tilde{\boldsymbol{x}}}\right)^{-1} = \frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{r}} = \frac{\mathrm{d}\left(\boldsymbol{x}+\boldsymbol{r}\right)}{\mathrm{d}\boldsymbol{r}} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{r}} + \boldsymbol{I}_{\mathrm{u}} = \left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\boldsymbol{x}}\right)^{-1} + \boldsymbol{I}_{\mathrm{u}}, \tag{14}$$

with $\boldsymbol{I}_{\mathrm{u}}$ the u-dimensional identity matrix. Furthermore, the same relation applies for the approximations

$$\widetilde{\boldsymbol{N}}^k = \boldsymbol{N}^k + \boldsymbol{I}_{\mathrm{u}}, \tag{15}$$

where $\widetilde{\boldsymbol{N}}^k \in \mathbb{R}^{\mathrm{u}\times\mathrm{u}}$ is the estimation of $\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\tilde{\boldsymbol{x}}}\right)^{-1}$. This is discussed further in Section 2.2. The two main methods which use this iteration scheme are IQN-ILS and IQN-MVJ.

Block Newton-Raphson iterations

Instead of regarding the problem as the search for the root of a single non-linear equation, one may solve Eq. (5) for both $\boldsymbol{x}$ and $\boldsymbol{y}$ by performing block Newton-Raphson iterations as follows

$$\begin{bmatrix} \boldsymbol{\mathcal{F}}'(\boldsymbol{x}) & -\boldsymbol{I}_{\mathrm{w}} \\ -\boldsymbol{I}_{\mathrm{u}} & \boldsymbol{\mathcal{S}}'(\boldsymbol{y}) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\mathcal{F}}(\boldsymbol{x}) - \boldsymbol{y} \\ \boldsymbol{\mathcal{S}}(\boldsymbol{y}) - \boldsymbol{x} \end{bmatrix}, \tag{16}$$

where $\boldsymbol{\mathcal{F}}'(\boldsymbol{x}) = \frac{\mathrm{d}\tilde{\boldsymbol{y}}}{\mathrm{d}\boldsymbol{x}}$ and $\boldsymbol{\mathcal{S}}'(\boldsymbol{y}) = \frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{y}}$ denote the Jacobians of the flow and structure solver, respectively. These Jacobians are used to find a better guess for the input of each solver. A basic schematic is shown

in Figure 1c. The output of each solver is modified before being transferred to the next one. Again, the actual Jacobians are not available due to the black-box characteristics of the solvers, so approximations will be used, denoted by $\boldsymbol{M}_f^k \in \mathbb{R}^{\mathrm{w}\times\mathrm{u}}$ and $\boldsymbol{M}_s^k \in \mathbb{R}^{\mathrm{u}\times\mathrm{w}}$ for the flow and structural solver respectively.

At the start of an iteration for instance, the input of the flow solver $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \Delta\boldsymbol{x}^k$ will be calculated by solving the system

$$\left(\boldsymbol{I}_\mathrm{u} - \boldsymbol{M}_s^k\boldsymbol{M}_f^k\right)\Delta\boldsymbol{x}^k = \tilde{\boldsymbol{x}}^k - \boldsymbol{x}^k + \boldsymbol{M}_s^k(\tilde{\boldsymbol{y}}^k - \boldsymbol{y}^k) \tag{17}$$

for $\Delta\boldsymbol{x}^k$. In this formula $\boldsymbol{x}^k$ and $\tilde{\boldsymbol{y}}^k$ are the in- and output of the flow solver in the previous iteration and $\boldsymbol{y}^k$ and $\tilde{\boldsymbol{x}}^k$ are analogously the in- and output of the structural solver. A good way to solve this linear system is to use an iterative Krylov solver, such as the generalize minimal residual (GMRES) method [13]. As such, the explicit construction of the Jacobian matrices is avoided, as it is sufficient to be able to calculate the product of a Jacobian matrix with a vector. A matrix-free procedure is obtained. Analogously, the input $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \Delta\boldsymbol{y}^k$ for the structural solver is acquired by solving

$$\left(\boldsymbol{I}_\mathrm{w} - \boldsymbol{M}_f^{k+1}\boldsymbol{M}_s^k\right)\Delta\boldsymbol{y}^k = \tilde{\boldsymbol{y}}^{k+1} - \boldsymbol{y}^k + \boldsymbol{M}_f^{k+1}(\tilde{\boldsymbol{x}}^k - \boldsymbol{x}^{k+1}) \tag{18}$$

for $\Delta\boldsymbol{y}^k$. Because the coefficient matrices in Eq. (17) and Eq. (18) are already full rank, there is no need to apply the approximation methods on other Jacobians as was the case with the residual operator scheme. Furthermore, in this iteration scheme, two approximate Jacobians are constructed in each iteration and the Jacobian itself is required, rather than its inverse. This iteration scheme is used in IBQN-LS and MVQN.

2.2 Two methods for constructing the approximate Jacobians

The previously discussed iteration schemes, meant to improve the stability of Gauss-Seidel iterations, require the availability of one or multiple (inverse) Jacobians. The residual operator scheme uses the inverse Jacobian $\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\tilde{\boldsymbol{x}}}\right)^{-1} = \frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{r}}$, whereas in the block iteration scheme, the Jacobians $\frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{y}}$ and $\frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{y}}$ are needed. However, as the solvers are treated as black-boxes, these Jacobians are not accessible and have to be approximated. Previous research showed that the instabilities in Gauss-Seidel iterations are caused by a limited set of modes [5, 6]. As a consequence, the full Jacobian is not needed and a low-rank approximation suffices. Taking advantage of this observation, the required Jacobian is constructed based on in- and outputs from previous solver evaluations. To this end, differences between vectors from consecutive iterations are stored as columns of two matrices: $\boldsymbol{V}$ and $\boldsymbol{W}$.

In this section, the procedure will be explained for the residual operator scheme, i.e. the approximation of $\frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{r}}$ by $\widetilde{\boldsymbol{N}}^k$. For this scheme, the method receives an input $\Delta\boldsymbol{r}^k$ and returns an estimation $\widetilde{\boldsymbol{N}}^k\Delta\boldsymbol{r}^k$ of $\Delta\tilde{\boldsymbol{x}}^k$. Therefore, inputs $\boldsymbol{r}$ and outputs $\tilde{\boldsymbol{x}}$ from previous iterations are stored in the matrices $\boldsymbol{V}$ and $\boldsymbol{W}$, respectively. The same approximation methods can equally be applied to the block iteration scheme, by replacing $\boldsymbol{r}$ and $\tilde{\boldsymbol{x}}$ with the correct in- and outputs. The vectors $\boldsymbol{x}$ and $\tilde{\boldsymbol{y}}$ are the in- and output for the approximation $\boldsymbol{M}_f^k$ of $\frac{\mathrm{d}\tilde{\boldsymbol{y}}}{\mathrm{d}\boldsymbol{x}}$. Likewise, $\boldsymbol{y}$ and $\tilde{\boldsymbol{x}}$ are the in- and output for the approximation $\boldsymbol{M}_s^k$ of $\frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{y}}$.

Returning to the residual operator scheme, the set of differences between consecutive iterations $\delta\boldsymbol{r}^i = \boldsymbol{r}^{i+1} - \boldsymbol{r}^i$ and $\delta\tilde{\boldsymbol{x}}^i = \tilde{\boldsymbol{x}}^{i+1} - \tilde{\boldsymbol{x}}^i$ for $i = 0, \ldots, k-1$ are stored after each iteration k of the current time step $n+1$, as columns of the matrices $\boldsymbol{V}^{n+1,k}$ and $\boldsymbol{W}^{n+1,k}$, respectively.

$$\begin{aligned}\boldsymbol{V}^{n+1,k} &= \begin{bmatrix}\delta\boldsymbol{r}^{k-1} & \delta\boldsymbol{r}^{k-2} & \ldots & \delta\boldsymbol{r}^1 & \delta\boldsymbol{r}^0\end{bmatrix}\\ \boldsymbol{W}^{n+1,k} &= \begin{bmatrix}\delta\tilde{\boldsymbol{x}}^{k-1} & \delta\tilde{\boldsymbol{x}}^{k-2} & \ldots & \delta\tilde{\boldsymbol{x}}^1 & \delta\tilde{\boldsymbol{x}}^0\end{bmatrix}\end{aligned} \tag{19}$$

For brevity, the index $n+1$, indicating the current time step, will be left out if no confusion can occur. The number of columns of $\boldsymbol{V}^k$ and $\boldsymbol{W}^k$ is indicated with v and is typically much larger than the number of rows u. Each difference $\delta\boldsymbol{r}^i$ has a one-to-one correspondence to a difference $\delta\tilde{\boldsymbol{x}}^i$. It is postulated that the approximated Jacobian $\widetilde{\boldsymbol{N}}^k$ has to fulfill each of these k relations

$$\delta\tilde{\boldsymbol{x}}^i = \widetilde{\boldsymbol{N}}^k \delta\boldsymbol{r}^i, \tag{20}$$

further called the secant equations. This can concisely be expressed as $\boldsymbol{W}^k = \widetilde{\boldsymbol{N}}^k \boldsymbol{V}^k$.

The convergence speed is greatly increased if information from previous time steps is included, due to similarity between subsequent time steps. The manner in which this is achieved, is different for the two approximation methods discussed in this section. However, without reuse, both are identical. Therefore, the approach without reuse will be explained first.

Typically, after the first iteration insufficient information is available to construct an approximate Jacobian, such that it is approximated as $-\boldsymbol{I}_{\mathrm{u}}$.

Satisfying the secant equations can be regarded as writing the input $\Delta\boldsymbol{r}^k$ as a linear combination of the known differences $\delta\boldsymbol{r}^i$, and composing the required $\Delta\tilde{\boldsymbol{x}}$ from the corresponding differences $\delta\tilde{\boldsymbol{x}}^i$. This linearization of $\Delta\boldsymbol{r}$ can be written as

$$\Delta\boldsymbol{r}^k \approx \boldsymbol{V}^k \boldsymbol{c}^k \tag{21}$$

with $\boldsymbol{c}^k \in \mathbb{R}^{\mathrm{v}\times 1}$ the coefficients of the decomposition and v the number of columns of $\boldsymbol{V}^k$. Imposing a least-squares requirement to this overdetermined set of equations ($\mathrm{v} \leq \mathrm{u}$), results in a unique solution given by

$$\boldsymbol{c}^k = ({\boldsymbol{V}^k}^{\mathrm{T}} \boldsymbol{V}^k)^{-1} {\boldsymbol{V}^k}^{\mathrm{T}} \Delta\boldsymbol{r}^k. \tag{22}$$

However, this implementation becomes rather unstable if the number of columns in $\boldsymbol{V}^k$ becomes high. A better approach is to calculate the economy-size QR-decomposition using Householder transformations [14].

$$\boldsymbol{V}^k = \boldsymbol{Q}^k \boldsymbol{R}^k \tag{23}$$

Note that $\boldsymbol{Q}^k$ has orthonormal columns such that ${\boldsymbol{Q}^k}^{\mathrm{T}} \boldsymbol{Q}^k = \boldsymbol{I}_{\mathrm{u}}$, but because $\boldsymbol{Q}^k$ is rectangular $\boldsymbol{Q}^k {\boldsymbol{Q}^k}^{\mathrm{T}} \neq \boldsymbol{I}_{\mathrm{v}}$. The coefficient vector $\boldsymbol{c}^k$ is then calculated by solving the triangular system

$$\boldsymbol{R}^k \boldsymbol{c}^k = {\boldsymbol{Q}^k}^{\mathrm{T}} \Delta\boldsymbol{r}^k \tag{24}$$

using back substitution. Finally, the resulting $\Delta\tilde{\boldsymbol{x}}$ is calculated as

$$\Delta\tilde{\boldsymbol{x}}^k = \boldsymbol{W}^k \boldsymbol{c}^k, \tag{25}$$

which means that $\Delta\tilde{\boldsymbol{x}}^k$ is assumed to consist of the same linear combination of old modes as $\Delta\boldsymbol{r}^k$. As such, the approximate Jacobian is symbolically given by

$$\boldsymbol{N}^k = \widetilde{\boldsymbol{N}}^k - \boldsymbol{I}_{\mathrm{u}} = \boldsymbol{W}^k {\boldsymbol{R}^k}^{-1} {\boldsymbol{Q}^k}^{\mathrm{T}} - \boldsymbol{I}_{\mathrm{u}}. \tag{26}$$

However, this matrix should never be constructed explicitly, as only the product of this matrix with an arbitrary vector is required (Eq. (13)). As such, the procedure is matrix-free, in the sense that no large dense matrices are constructed.

Note that the Jacobian $\frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}\boldsymbol{r}}$ is a square matrix with a dimension u equal to the vector dimension d multiplied by the number of grid points on the interface. This is normally much larger than the number of secant equations k. Hence, in order to uniquely define its approximation, a least-squares requirement is applied.

Least-squares approach to reuse

A first possibility to include information from previous time steps, is expanding the matrices $\boldsymbol{V}^k$ and $\boldsymbol{W}^k$ by including those from the q previous time steps.

$$\begin{aligned} \boldsymbol{V}^k &= \begin{bmatrix} \boldsymbol{V}^{n+1,k} & \boldsymbol{V}^n & \dots & \boldsymbol{V}^{n+1-q} \end{bmatrix} \\ \boldsymbol{W}^k &= \begin{bmatrix} \boldsymbol{W}^{n+1,k} & \boldsymbol{W}^n & \dots & \boldsymbol{W}^{n+1-q} \end{bmatrix} \end{aligned} \tag{27}$$

Note that the resulting matrices are denoted with $\boldsymbol{V}^k$ and $\boldsymbol{W}^k$, such that the formulas stated above directly apply. Again, a least-squares requirement is imposed to uniquely define the matrix and all secant equations are satisfied, also those from previous time steps.

Due to the increase in number of columns, it becomes important to remove columns from $\boldsymbol{V}^k$ which are (almost) linear dependent, in order to be able to solve system Eq. (24). Therefore, an appropriate filtering has to be applied [15, 16]. After calculation of the QR-decomposition, the diagonal elements of $\boldsymbol{R}^k$, starting from the left, are compared with a tolerance ϵ. When the value is too low, the corresponding column in $\boldsymbol{V}^k$ and $\boldsymbol{W}^k$ is removed. Then, the QR-decomposition is recalculated. This is repeated until all diagonal elements in $\boldsymbol{R}^k$ are higher than ϵ. As it is beneficial to retain the most recent information, new columns are added on the left. This is also the reason why the QR-decomposition cannot be updated, but has to be recalculated in every iteration. However, the associated computational cost is typically negligible compared to the cost of the solvers. If the number of degrees of freedom on the interface u is low, possibly the outer most right columns have to be removed as well, to achieve $\mathrm{v} \leq \mathrm{u}$.

The optimal number of time steps which should be taken into account is problem dependent, but usually varies around $\boldsymbol{q} \approx 10$. If $\boldsymbol{q}$ is too low, useful information is disregarded, but when $\boldsymbol{q}$ is too high irrelevant information is taken into account, which corrupts the validity of the approximation of the Jacobian. Note that the method remains matrix-free, even after the introduction of reuse. Quasi-Newton techniques using this type of approach to approximate the Jacobian are IQN-ILS and IBQN-LS.

Multi-vector approach to reuse

A second approach to find a suitable approximation for the Jacobian is the so called multi-vector technique. Besides fulfilling the secant equations in the current time step,

$$\boldsymbol{W}^{n+1,k} = \widetilde{\boldsymbol{N}}^{n+1,k} \boldsymbol{V}^{n+1,k}, \tag{28}$$

this method requires the approximate Jacobian with reuse from the current time step $\widetilde{\boldsymbol{N}}^{n+1,k}$ to be as close as possible to the approximate Jacobian from the previous one $\widetilde{\boldsymbol{N}}^{n}$ or, in other words, a least-squares approach on the difference with the previous Jacobian is used. Remark that not applying reuse, corresponds to setting the previous Jacobian $\widetilde{\boldsymbol{N}}^{n}$ to zero in every iteration, which boils down to the method explained in the beginning of this section. If the current Jacobian is split in a part due to the previous time step and one due to the current time step as follows

$$\widetilde{\boldsymbol{N}}^{n+1,k} = \widetilde{\boldsymbol{N}}^{n} + \Delta\widetilde{\boldsymbol{N}}^{n+1,k}, \tag{29}$$

Eq. (20) becomes

$$\Delta\widetilde{\boldsymbol{N}}^{n+1,k}\boldsymbol{V}^{n+1,k} = \boldsymbol{W}^{n+1,k} - \widetilde{\boldsymbol{N}}^{n}\boldsymbol{V}^{n+1,k}. \tag{30}$$

Applying a least square condition on Eq. (30) results in the following update formula

$$\widetilde{\boldsymbol{N}}^{n+1,k} = \widetilde{\boldsymbol{N}}^{n} + \left(\boldsymbol{W}^{n+1,k} - \widetilde{\boldsymbol{N}}^{n}\boldsymbol{V}^{n+1,k}\right)\left(\left(\boldsymbol{V}^{n+1,k}\right)^{\mathrm{T}}\boldsymbol{V}^{n+1,k}\right)^{-1}\left(\boldsymbol{V}^{n+1,k}\right)^{\mathrm{T}}. \tag{31}$$

Simply put, no change of the Jacobian is allowed in directions that have not been encountered in the current time step (no change condition) and old information from previous time steps is replaced by newer information when it becomes available.

This method was devised to avoid the determination of the problem dependent parameter $\boldsymbol{q}$. The information from previous time steps is taken into account implicitly and is removed automatically when new information becomes available. As the number of columns of $\boldsymbol{V}^k$ remains small, there is less need for filtering.

The main disadvantage of this method is that it requires the explicit construction of the Jacobian, which leads to increased memory requirements and calculation speed. A matrix-free method has a memory requirement proportional to the number of rows u and columns v compared to the explicit matrix construction, where the memory requirements scale with u^2. Especially, simulations with a high number of degrees of freedom u on the fluid-structure interface favor a matrix-free method. Alternative, matrix-free implementations of the multi-vector approach have been proposed, but these reintroduce a reuse parameter [17, 18]. The methods using the multi-vector approach are IQN-MVJ and MVQN.

3 TEST CASE

The four quasi-Newton techniques presented above are now tested on a one-dimensional (1D) test case.

3.1 One-dimensional flexible tube

Consider a straight flexible tube of length ℓ with circular cross-section with nominal diameter r_0 and wall thickness h. The fluid has a density ρ_f. The structure has a modulus of elasticity E, a Poisson's ratio ν and density ρ_s. The parameter values are included in Table 1.

The flow is governed by the continuity and momentum equation in which viscosity and gravity are neglected:

$$\begin{aligned} \frac{\partial a}{\partial t} + \frac{\partial av}{\partial z} &= 0 \\ \frac{\partial av}{\partial t} + \frac{\partial av^2}{\partial z} + \frac{1}{\rho_f}\left(\frac{\partial ap}{\partial z} - p\frac{\partial a}{\partial z}\right) &= 0 \end{aligned} \tag{32}$$

with $a = \pi r^2$ the cross-sectional area and r the inner radius of the tube. Furthermore, t, v, p are the time, the velocity along the axis of the tube and the pressure, respectively. On the outlet of the fluid domain a fixed pressure equal to the reference pressure $p_0 = 0\ Pa$ is specified. On the inlet a pressure pulse is imposed with amplitude 1333.2 Pa and a duration of 0.003 s.

The deformation of the tube wall is governed by

$$\rho_s h\frac{\partial^2 r}{\partial t^2} + b_1\frac{\partial^4 r}{\partial z^4} - b_2\frac{\partial^2 r}{\partial z^2} + b_3(r - r_o) = p - p_o. \tag{33}$$

Table 1: Parameter values of the 1D flexible tube case

Parameter	Value	Parameter	Value
ℓ	0.05 m	E	300 000 $kg/m\,s^2$
r_0	0.005 m	ν	0.3
h	0.001 m	ρ_f	1 000 kg/m^3
Δt	0.0001 s	ρ_s	1 200 kg/m^3

where parameters b_1, b_2 and b_3 account for stresses and bending in the wall [19, 20]. Only radial displacements are considered, so the length of the tube stays constant. The nominal radius r_o corresponds to a uniform pressure p_o if the structure is at rest. Here, both tube ends are clamped and the wall is considered thin. The values of b_1, b_2 and b_3 are then calculated as

$$b_1 = \frac{hE}{1-\nu^2}\frac{h^2}{12} \qquad b_2 = \frac{hE}{1-\nu^2}\frac{h^2}{12}\frac{2\nu}{r_o^2} \qquad b_3 = \frac{hE}{1-\nu^2}\frac{1}{r_o^2} + \frac{hE}{1-\nu^2}\frac{h^2}{12}\frac{1}{r_o^4} \tag{34}$$

Because $h \ll r_o$, the second term of b_3 is neglected with respect to the first one.

The tube is discretized on a 1D grid with m cells with a width equal to $\Delta z = \ell/m$. The test case above is solved for t from 0 to 0.01 s, in 100 time steps Δt of 0.0001 s.

3.2 CoCoNuT

The FSI-problem is solved with the open-source code *CoCoNuT*, developed in-house. This code offers an easy way to couple different software packages, which are treated as black-boxes. It has the advantage of being modular and flexible in combining mappers, solvers and coupling algorithms. Moreover, due to its comprehensive structure and straightforward implementation with Python, the code is easily adapted to users own requirements, if needed. The most recent code can be found on GitHub in the repository pyfsi/coconut. A frozen version used to generate the results in this paper is located on fsi.ugent.be. Both a flow and structure solver were written for the 1D flexible tube case described in Section 3.1, the code of which is equally available online.

3.3 Results

As the solvers are typically very expensive in FSI-calculations, the focus is on the number of solver executions. The term iteration will be used to denote a subsequent flow and structure solver call. Table 2 shows the number of iterations for the different quasi-Newton methods discussed above, as well as with simple relaxation and Aitken relaxation. The convergence tolerance ϵ_x is 10^{-12}. Dynamic relaxation improves convergence greatly compared to simple relaxation, but does not perform as well as the quasi-Newton methods. The block and residual operator methods perform similarly, whereas the multi-vector techniques turn out to be somewhat more efficient compared to the least-squares techniques for this test case. Note that the latter indeed perform best for q equal to 10. IQN-ILS (q=0) corresponds to the method without reuse. This shows the importance of reusing information from previous time steps.

Next, Table 3 shows the memory requirements, for different number of cells m. The reported values are the maximum of the used memory after each iteration as determined using the guppy3 module in Python. The results clearly show quadratically scaling memory requirements for the multi-vector techniques, in contrast to the linearly scaling requirements for the least-squares techniques, due to their

Table 2: Average number of iteration required per time step for the different methods.

Method	Number of iterations	Method	Number of iterations
Relaxation (ω=0.05)	287.95	Aitken relaxation	25.49
IQN-ILS (q=0)	10.90	IBQN-LS (q=0)	10.80
IQN-ILS (q=1)	8.27	IBQN-LS (q=1)	8.38
IQN-ILS (q=5)	5.92	IBQN-LS (q=5)	6.04
IQN-ILS (q=10)	5.18	IBQN-LS (q=10)	5.32
IQN-ILS (q=20)	5.87	IBQN-LS (q=20)	5.74
IQN-MVJ	4.27	MVQN	4.46

matrix-free implementation. Furthermore, the block methods require more memory compared to the residual operator techniques, as two Jacobian matrices are approximated.

Table 3: Memory requirements for the different methods in MB.

Method	m=100	m=1 000	m=10 000
IQN-ILS (q=10)	18	32	169
IBQN-LS (q=10)	20	39	224
IQN-MVJ	19	175	14 556
MVQN	24	422	38 623

4 CONCLUSION

For cases with high-added mass, Gauss-Seidel iterations are not stable or converge very slowly even when relaxation is applied. In these cases quasi-Newton methods lead to an important improvement in stability and convergence speed. Four different quasi-Newton techniques were discussed and categorized based upon iteration scheme and Jacobian approximation. While the block methods require two Jacobian approximations, only one is needed for the residual operator methods. The least-squares approach can be implemented matrix-free which results in linearly scaling memory requirements, whereas for the multi-vector technique, memory requirements scale quadratically, due to the explicit matrix construction. This is especially bad for problems with a high number of degrees of freedom on the interface. Tests were performed on a 1D flexible tube using the Python coupling tool *CoCoNuT*. In terms of convergence the quasi-Newton methods clearly outperform Aitken relaxation, but are comparable between themselves.

REFERENCES

[1] Schott, B., Ager, C. and Wall., W.A. A monolithic approach to fluid-structure interaction based on a hybrid Eulerian-ALE fluid domain decomposition involving cut elements. *Int. J. Numer. Methods. Eng.* (2019) **119(3)**:208–237.

[2] Farhat C., van der Zee,K.G. and Geuzaine, P.. Provably second-order timeaccurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Comput. Methods. Appl. Mech. Eng.* (2006) **195(17-18)**:1973–2001.

[3] van Brummelen, E.H. Added mass effects of compressible and incompressible flows in fluid-structure interaction. *J. Appl. Mech.* (2009) **76(2)**:021206-1–7.

[4] Causin, P., Gerbeau, J.-F., and Nobile. F. Added-mass effect in the design of partitioned algorithms for fluid–structure problems. *Comput. Methods. Appl. Mech. Eng.* (2005) **194(42-44)**:4506–4527.

[5] Degroote, J., Bruggeman, P., Haelterman, R. and Vierendeels, J. Stability of a coupling technique for partitioned solvers in FSI applications. *Comput. Struct.* (2008) **86(23-24)**:2224–2234.

[6] Degroote, J., Annerel, S. and Vierendeels, J. Stability analysis of Gauss–Seidel iterations in a partitioned simulation of fluid–structure interaction. *Comput. Struct.* (2010) **88(5–6)**:263–271.

[7] Vierendeels, J., Lanoye, L., Degroote, J. and Verdonck, P.R. Implicit coupling of partitioned fluid-structure interaction problems with reduced order models. *Comput. Struct.* (2007) **85(11-14)**:970–976.

[8] Degroote, J., Bathe, K.-J. and Vierendeels, J. Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction. *Comput. Struct.* (2009) **87(11-12)**:793–801.

[9] Bogaers A.E.J., Kok, S., Reddy, B.D., and Franz, T. Quasi-Newton methods for implicit black-box FSI coupling. *Comput. Methods Appl. Mech. Eng.* (2014) **279**:113–132.

[10] Lindner, F. and Mehl, M. and Scheufele, K. and Uekermann, B. A comparison of various quasi-Newton schemes for partitioned fluid-structure interaction. *6th Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering*. (2015) 477–488.

[11] Blom, D., Lindner, F., Mehl, M., Scheufele, K., Uekermann, B., and van Zuijlen, A. A review on fast quasi-Newton and accelerated fixed-point iterations for partitioned fluid-structure interaction simulation. In *Advances in Computational Fluid-Structure Interaction and Flow Simulation, Modeling and Simulation in Science, Engineering and Technology*, pages 257–269. Birkhäuser, Cham, (2016).

[12] Küttler, U. and Wall, W.A. Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Comput. Mech.* (2008) **43(1)**:61–72.

[13] Degroote, J., Haelterman, R., Annerel, S., Bruggeman, P. and Vierendeels J. Performance of partitioned procedures in fluid–structure interaction. *Comput. Struct.* (2010) **88(7-8)**:446–457.

[14] Trefethen, L.N. and Bau, D. *Numerical Linear Algebra*. SIAM, (1997).

[15] Degroote, J. and Vierendeels, J. Multi-level quasi-Newton coupling algorithms for the partitioned simulation of fluid-structure interaction. *Comput. Methods Appl. Mech. Eng.* (2012) **225–228**:14–27.

[16] Haelterman, R., Bogaers, A.E.J., Scheufele, K., Uekermann, B. and Mehl., M. Improving the performance of the partitioned QN-ILS procedure for fluid–structure interaction problems: Filtering. *Comput. Struct.* (2016) **171**:9–17.

[17] Scheufele, K.and Mehl., M. Robust Multisecant Quasi-Newton Variants for Parallel Fluid-Structure Simulations—and Other Multiphysics Applications. *SIAM J. Sci. Comput.* (2017) **39(5)**:S404–S433.

[18] Spenke, T., Hosters, N. and Behr., M. A multi-vector interface quasi-Newton method with linear complexity for partitioned fluid–structure interaction. *Comput. Methods Appl. Mech. Eng.* (2020) **361**:112810.

[19] Quarteroni, A. and Tuveri, M. and Veneziani, A. Computational vascular fluid dynamics: problems, models and methods. *Comput. Vis. Sci.* (2000) **2(4)**:163–197.

[20] Čanić, S., Mikelić, A. and Tambača, J. A two-dimensional effective model describing fluid–structure interaction in blood flow: analysis, simulation and experimental validation. *C. R. Mécanique*. (2005) **333(12)**:867–883.