Mathematical and Computational Applications

Article

Enhancing Quasi-Newton Acceleration for Fluid-Structure Interaction

Kyle Davis [1,*], Miriam Schulte [1] and Benjamin Uekermann [2]

[1] Simulation of Large Systems, Institute for Parallel and Distributed Systems (IPVS), University of Stuttgart, 70569 Stuttgart, Germany; miriam.schulte@ipvs.uni-stuttgart.de

[2] Usability and Sustainability of Simulation Software, Institute for Parallel and Distributed Systems (IPVS), University of Stuttgart, 70569 Stuttgart, Germany; benjamin.uekermann@ipvs.uni-stuttgart.de

* Correspondence: kyle.davis@ipvs.uni-stuttgart.de

Abstract: We propose two enhancements of quasi-Newton methods used to accelerate coupling iterations for partitioned fluid-structure interaction. Quasi-Newton methods have been established as flexible, yet robust, efficient and accurate coupling methods of multi-physics simulations in general. The coupling library preCICE provides several variants, the so-called IQN-ILS method being the most commonly used. It uses input and output differences of the coupled solvers collected in previous iterations and time steps to approximate Newton iterations. To make quasi-Newton methods both applicable for parallel coupling (where these differences contain data from different physical fields) and to provide a robust approach for re-using information, a combination of information filtering and scaling for the different physical fields is typically required. This leads to good convergence, but increases the cost per iteration. We propose two new approaches—pre-scaling weight monitoring and a new, so-called QR3 filter, to substantially improve runtime while not affecting convergence quality. We evaluate these for a variety of fluid-structure interaction examples. Results show that we achieve drastic speedups for the pure quasi-Newton update steps. In the future, we intend to apply the methods also to volume-coupled scenarios, where these gains can be decisive for the feasibility of the coupling approach.

Keywords: fluid-structure interaction; quasi-Newton; multiphysics coupling

check for updates

Citation: Davis, K.; Schulte, M.; Uekermann, B. Enhancing Quasi-Newton Acceleration for Fluid-Structure Interaction. *Math. Comput. Appl.* **2022**, *27*, 40. https://doi.org/10.3390/mca27030040

Academic Editors: Simona Perotto, Gianluigi Rozza and Antonia Larese

Received: 16 February 2022
Accepted: 27 April 2022
Published: 6 May 2022

1. Introduction

Multiphysics simulations have shown immense usefulness in the engineering design sector, and are increasingly being applied to more complex problems, ranging from biomedical devices [1] and wind loads on structures [2], to hydraulic fracture simulation [3]. The rise in challenging applications of multiphysics simulations has led to an increased focus on developing flexible, efficient, and scalable multiphysics coupling software. A practical and user friendly approach is to develop partitioned-coupling software, which couples existing standalone physics simulation solvers together to solve new types of simulation problems. Here, the physics solvers themselves are treated as black boxes. This is in contrast to monolithic methods, where all of the equations from each physics domain are solved together in a single system. Partitioned coupling requires an additional piece of software taking care of the actual numerical and technical coupling of the separate solvers. Amongst various such coupling software packages available is preCICE [4]. The key features of preCICE are the minimally invasive library approach, sophisticated numerical methods, parallel scalability, and a strong focus on usability, maintainability, and extensibility. In this paper, we present enhancements and robust parameter choices for numerical equation coupling with preCICE, substantially improving performance, robustness and usability.

Various other general-purpose coupling software exist that are able to perform partitioned coupling for multiphysics (including fluid-structure interaction) and multi-scale problems. Software coupling packages similar to preCICE are DTK [5] and

OpenPALM [6], which both offer a slightly different approach to simulation coupling than preCICE. DTK's application programming interface (API) offers lower-level features compared to preCICE, allowing more flexibility regarding the coupling logic, but at a greater development effort for the user, whereas OpenPALM offers a higher-level approach, with built-in coupling logic and a graphical user interface. A comparison of preCICE with the mentioned libraries, as well as many others coupling software solutions, is provided in [7].

In addition to numerical coupling of separate solvers, a coupling software has to provide communication between solvers, data mapping between non-matching meshes at the interface between solvers, and interpolation in time (if higher order time stepping shall be achieved). We focus on the numerical coupling in this paper, with a specific focus on parallel quasi-Newton schemes. Quasi-Newton schemes for partitioned multi-physics coupling were introduced in 2009 [8] and have been improved since [9–13]. Similar methods have been developed in a different community in the context of acceleration of fixed-point solvers under the name Anderson mixing or Anderson acceleration [14–17]. Quasi-Newton acceleration schemes have been shown to provide fast and stable coupling between various physics solvers for a variety of problems [3,9,18]. However, to provide fast and stable coupling, additional numerical techniques are implemented in preCICE [10]. These additional interface operations have often been considered to have a negligible computational cost compared to the solvers, as the interface degrees of freedom are assumed to be much fewer than those of the coupled solvers themselves. However, this might not be true if the interface becomes large or in the case of volume coupling, where the entire domain is essentially the coupling interface.

The aim of this work is to implement minor enhancements to the existing implementation of quasi-Newton methods in preCICE, which, however, give major improvements in terms of efficiency, robustness, and usability. The enhancements allow easier access to good input parameter choices for standard users, while also offering maximal flexibility for expert users without having to consider the computational cost of the coupling library. We approach this goal by achieving a detailed understanding on how the implementation of quasi-Newton schemes affects computational performance.

The remainder of the paper is structured as follows. In Section 2, we provide an overview of multisecant quasi-Newton methods for fixed-point problems, as well as general existing methods used to improve the quasi-Newton performance in preCICE. In Section 3, we show how these additional general methods can be further enhanced to reduce the number of coupling iterations, on the one hand, and the runtime of the actual acceleration, on the other hand. In Section 4, we present different test cases to analyse the improvements. In Section 5, we present results for these test cases, followed by a discussion of the results in Section 6.

2. Methods—Introduction to Quasi-Newton Coupling

In this section, we present variants of iterative quasi-Newton coupling as implemented for multiphysics simulation coupling. We start with the different versions of fixed-point equations that are solved in iterative partitioned coupling approaches in Section 2.1. In Section 2.2, we present the basic ideas of the quasi-Newton approaches used, followed by enhancements of these methods improving convergence and robustness in Section 2.3. The contributions of this paper, enhancements and rules that further improve the efficiency, robustness and usability of quasi-Newton coupling for multiphysics simulations, are presented in Section 3.

2.1. Partitioned Coupling

For time-dependent problems, partitioned coupling, i.e., coupling of separate simulation codes, can be divided into two types: explicitly (loosely) coupled or implicitly (strongly) coupled. In explicit coupling, each solver performs its time step only once and proceeds with the next time step after exchanging data with the other solvers. In implicit coupling, all solvers iterate their time steps exchanging data after each iteration until a fixed-point problem describing the coupling conditions is solved. Solving this fixed point problem requires the introduction of sophisticated methods, such as quasi-Newton methods, to achieve acceleration of the convergence of the respective fixed-point iterations.

To explain the derivation of the fixed-point formulation, we consider two coupled solvers for simplicity, represented by mapping functions S_1 and S_2 operating on data defined at the coupling interface Γ. A common example is a fluid-structure interaction (FSI) model, where S_1 is the fluid solver that maps interface displacements or velocities x_1 to forces x_2 exerted at the structure, whereas S_2 is the structure solver that maps forces x_2 to interface displacements or velocities x_1. The following description of the coupling, however, generalises to any multi-physics problem. The mapping S_1 requires the output of S_2 and vice-versa such that

$$S_1 : x_1 \mapsto x_2 \qquad \text{and} \qquad S_2 : x_2 \mapsto x_1.$$

For the considered time-dependent problems, x_1 and x_2 are the respective interface values at the new time step. Strong or implicit coupling between S_1 and S_2 is formulated as a fixed point problem. Two mathematically equivalent variants of this fixed-point problem can be formulated in matrix-like notation as

$$x_1 = S_2 \circ S_1(x_1) \text{ (Gauss-Seidel type coupling) and} \tag{1}$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & S_2 \\ S_1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ (Jacobi type coupling).} \tag{2}$$

These correspond to two different types of fixed-point iterations as depicted in Figure 1. In this work, we focus on the second, so-called Jacobi type system as it results in higher parallelism and, thus, more efficient usage of computational resources on large compute clusters and supercomputers [11], and allows for an arbitrary number of solvers to be coupled [19].

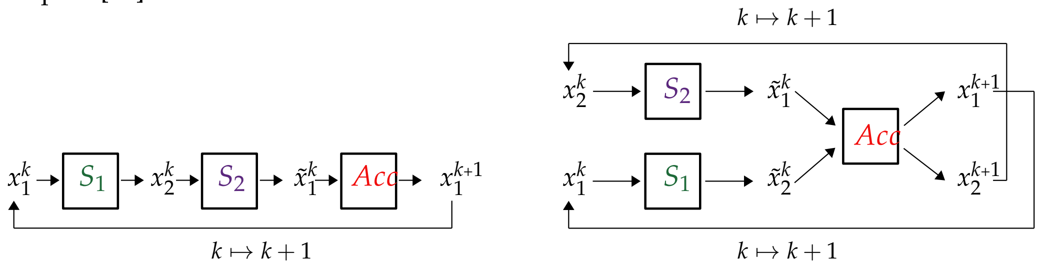


Figure 1. General coupling options, Gauss-Seidel (**left**) and Jacobi (**right**), for partitioned coupling of two solvers S_1 and S_2.

The unmodified fixed-point iterations $x^{k+1} = H(x^k)$, where

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ and } H = \begin{pmatrix} 0 & S_2 \\ S_1 & 0 \end{pmatrix} \tag{3}$$

in matrix-like notation, may be slow to converge or not converge at all [9]. We therefore apply an acceleration scheme, $Acc()$, such that

$$x^{k+1} = Acc(\tilde{x}^k) \text{ with } \tilde{x}^k = H(x^k).$$

In the following, we explain how the acceleration operator Acc is realised based on quasi-Newton approaches.

2.2. Introduction to Quasi-Newton Coupling Methods

The partitioned coupling schemes described above require solving a non-linear problem

$$R(x) := H(x) - x = 0. \tag{4}$$

To solve this for problem sizes typically encountered in multi-physics simulations, full Newton methods are undesirable due to large computational costs and memory requirements and, in addition, infeasible for black-box coupling as the derivatives of the fixed-point operator H are inaccessible. Quasi-Newton methods are able to approximate a Newton step and in general accelerate the convergence compared to the pure fixed-point iteration [8,9,12]. At each iteration, we compute the next iteration via

$$x^{k+1} = \tilde{x}^k - J^{-1} r^k, \tag{5}$$

where $r^k = R(x^k) = \tilde{x}^k - x^k$ and J^{-1} is an approximation of the inverse Jacobian of the mapping $\tilde{R} : \tilde{x} \mapsto \tilde{x} - H^{-1}(\tilde{x})$, i.e., the function that maps the result of the fixed point iteration to the residual ($\tilde{R}(H(x)) = R(x)$). Equation (5) represents a sequence of a fixed-point iteration and a modified approximated Newton step. As multiple iterations are performed in each time step for implicitly coupled systems, we introduce a convergence criteria to define when we proceed to the next time step. Implicitly coupled solvers are considered to have converged if

$$\frac{\|x_1^k - \tilde{x}_1^k\|_2}{\|x_1^k\|_2} < \epsilon_{\text{conv}}. \tag{6}$$

For solver S_1, and similarly for solver S_2. Critical to the performance of the quasi-Newton method is the manner in which we approach the approximation of the inverse Jacobian J^{-1}. We use the input/output differences collected from previous iterations collected in matrices

$$\begin{aligned} W_k^\eta &= \left[\Delta\tilde{x}^k, \Delta\tilde{x}^{k-1}, ..., \Delta\tilde{x}^{k-\eta}\right] \quad \text{with} \quad \Delta\tilde{x}^k = \tilde{x}^k - \tilde{x}^{k-1}, \\ V_k^\eta &= \left[\Delta r^k, \Delta r^{k-1}, ..., \Delta r^{k-\eta}\right] \quad \text{with} \quad \Delta r^k = r^k - r^{k-1}, \end{aligned} \tag{7}$$

where $W_k^\eta \in \mathbb{R}^{N\times\eta}$, $V_k^\eta \in \mathbb{R}^{N\times\eta}$, N is the number of degrees of freedom (DoF) at the coupling interface, and η is the maximum number of previous iterations that are retained. For transient coupled problems, this can also include iterations from previous time steps. We, however, drop iterations older than ζ time steps. The approximation of the inverse Jacobian is required to fulfil the multi-secant equation

$$J^{-1} V_k = W_k, \tag{8}$$

a strongly under-determined system as $\eta \ll N$. We, thus, have to add a norm minimisation

$$\min \|J^{-1} - J^{-1}_{\text{prev}}\|_F. \tag{9}$$

where J^{-1}_{prev} is a previous approximation. Depending on the choice of J^{-1}_{prev}, we get two types of quasi-Newton methods:

IQN-ILS. The Interface Quasi-Newton Inverse Least-Squares (IQN-ILS) method is a popular and frequently used multiphysics coupling acceleration scheme. It was first introduced in [8]. For IQN-ILS, we choose $J^{-1}_{prev} = 0$, i.e., to determine J^{-1}, we solve

$$J^{-1} V_k = W_k \;\text{ with } J^{-1} = \text{argmin}\|J^{-1}\|_F, \tag{10}$$

yielding

$$J^{-1} = W_k \left(V_k^T V_k\right)^{-1} V_k^T. \tag{11}$$

The benefit of this classic *least-squares* approach is the option for a matrix-free implementation of the quasi-Newton step (5):

$$x^{k+1} = \tilde{x}^k + W_k \alpha \;\text{ with }\; \alpha = \text{argmin}\|V_k \alpha + r^k\|_2. \tag{12}$$

To solve this least-squares problem, we compute a QR-decomposition of $V_k = QR$ and solve the small $\eta \times \eta$ system $R\alpha = -Q^T r^k$.

The IQN-ILS method builds the approximation of J^{-1} exclusively from the retained input/output vectors stored in W_k and V_k. Therefore, the amount and quality of the information stored in V_k and W_k is decisive for the convergence rate and robustness of the IQN-ILS method, and many problems typically require storing many previous iterations , η, over many previous time steps, ζ.

IQN-IMVJ. The Interface Quasi-Newton Inverse Multi-Vector Jacobian method [12,13,20] implicitly retains information from previous time steps using J_{prev}^{-1} as the previous time step's inverse Jacobian approximation and, thus, allows us to implicitly use information on J^{-1} already collected in previous time steps. In the IQN-IMVJ method, we also perform a QR-decomposition of $V_k^\eta = QR$ to determine the pseudo-inverse $V_k^\dagger = \left(V_k^T V_k\right)^{-1} V_k^T$ and get J^{-1} from

$$J^{-1} = J_{prev}^{-1} + \left(W_k - J_{prev}^{-1} V_k\right)\left(V_k^T V_k\right)^{-1} V_k^T. \tag{13}$$

The disadvantage of this method is that we need an explicit representation of J_{prev}^{-1} which requires $\mathcal{O}(N^2)$ both in memory and computational complexity. By smart approximations, this cost can, however, be reduced to $\mathcal{O}(N)$ [12,21]. In this paper, we focus on improvements to the IQN-ILS method, as the most commonly used quasi-Newton variant.

2.3. IQN-ILS Enhancement Techniques

When implementing the IQN-ILS method into a software library, various numerical methods need to be utilised to further improve numerical stability and reduce runtime and memory requirements. The two methods we highlight here are filtering of columns in V_k [10] and pre-scaling of the interface values [22]. These techniques are discussed below, and the advantages and disadvantages are highlighted.

2.3.1. Filtering

For IQN-ILS, there is no guarantee that all columns in V_k are linearly independent. Thus, to aid convergence for all acceleration schemes, filtering columns of the matrices V_k and W_k is performed to remove any linearly dependent columns [10]. We discuss two filtering variants: QR1 and QR2 filters (referred to as *old* QR and *new* QR filter, respectively, in Haeltemann et al. [10]). Both methods begin by performing a QR-decomposition $V_k = QR$, where V_k is decomposed into an orthogonal matrix $Q^{N\times\eta}$ and an upper triangular matrix $R^{\eta\times\eta}$.

In every iteration, a new column is added to the left of both V_k and W_k (see Equation (7)). Therefore the leftmost columns represent newest information and the rightmost columns older information. The QR-decomposition is performed column-wise from right to left, and is realised by a QR update procedure as shown in Figure 2. Taking an already computed decomposition QR from the previous iteration, a new column on the left $v = (V_k)_{:,1}$ (the subscript indicates the row and column number in a Python-like notation. $:,1$ refers to the leftmost column of V_k—this is the most recent column added to V_k) is orthogonalised against Q via a modified Gram–Schmidt procedure. The orthogonalised v is then added as additional (rightmost) column in Q (column q in Figure 2). A new (leftmost) column is also added to R (column r in Figure 2) along with a bottom row of zeros. A series of Givens rotations eliminate any non-zero sub-diagonal entries in R. A detailed explanation of the procedure can be found in [23]. Note that this seems to be unnecessarily complicated compared to adding columns from the right of V_k, which would only require orthogonalisation of the new column q with a standard Gram–Schmidt algorithm, and adding a new column r on the right of R. However, having the oldest information in the rightmost columns of V_k and W_k offers the advantage that deleting old information, possibly multiple columns at a time, is cheap. Deleting columns from V_k and W_k typically occurs at least every time step.

Deleting the rightmost (oldest) column of V_k only requires removing the rightmost column from Q and the rightmost column as well as the last row from R. This does not introduce non-zero sub-diagonal elements in R and, thus, does not require additional Givens rotations. Removing an arbitrary column from V_k requires (1) removing the corresponding column from R, (2) removing any sub-diagonal elements from R using Givens rotations, (3) applying the corresponding Givens-rotations to Q from the right, (4) removing the last column from Q as well as the bottom (zero) row from R.

If a complete QR decomposition is required, Q and R are discarded completely, and new Q and R matrices are rebuilt using each column from V_k, adding one column at a time.

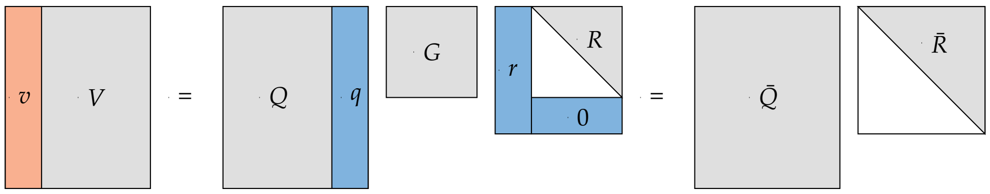


Figure 2. QR-decomposition updating procedure. For every new column added to V_k on the left, the QR decomposition can be updated by adding the orthogonalised version of this column as new (**rightmost**) column q in Q, adding an additional row of zeros to R and subsequently adding a new (**leftmost**) column r to R representing the respective orthogonalisation factors. To compute the final decomposition, we have to apply Given rotations (represented by the matrix G in the figure) to eliminate sub-diagonal entries in R. $\bar{Q}$ and $\bar{R}$ represent the final decomposition result.

QR1. In the QR1 filter, the impact of a column on the condition of R is estimated by comparing the diagonal elements of R to the complete norm of R as a metric for the norm of orthogonalised columns of V_k before normalisation. This means, we delete a column i if

$$R_{ii} < \epsilon_f \cdot \|R\|_F, \tag{14}$$

where ϵ_f is the filtering limit, a user specified parameter. This filter has a potential drawback as the QR-decomposition is built from the oldest column of information to the newest and, thus, has a tendency to delete new columns instead of (potentially outdated) old ones. However, [22] found that this was not problematic with a well selected filter limit. The main advantage of the QR1 filter is that it does not enforce a complete re-computation of the QR-decomposition in each coupling iteration, but allows updating the decomposition by inserting the newest column v into QR only. Any filtered column can be removed with a single column deletion step.

QR2. The QR2 filter (Algorithm 1) was introduced in [10] as a means to quantify the amount of new information a column in V_k adds to the QR-decomposition, and to filter the columns during the construction of Q and R itself. Importantly, the QR2 method re-constructs the QR-decomposition beginning with the newest column of information and, thus, has to rebuild the QR decomposition in each iteration, and tends to remove the oldest columns from V_k. This is favoured as older columns might no longer be relevant to the current dynamics of the physical system.

Algorithm 1 : QR2 Filter [10]

```
$R_{11} = \|(V_k)_{:,1}\|_2$                              ▷ This is the newest column
for $i = 1, ..., \eta$ do                                  ▷ Starts from newest column
    $\overline{v} = (V_k)_{:,i}$
    for $j = 1, ..., i-1$ do
        $R_{ji} = Q^T_{:,j} \cdot \overline{v}$
        $\overline{v} = \overline{v} - R_{ji} \cdot Q_{:,j}$
    end for
    if $\|\overline{v}\|_2 < \epsilon_f \|(V_k)_{:,i}\|_2$ then
        delete column $i$
    end if
    $R_{ii} = \|\overline{v}\|_2$ and $Q_{:,i} = \overline{v}/R_{ii}$
end for
```

2.3.2. Pre-Scaling

When using the Jacobi-type fixed point equation as introduced in Equation (2), largely different orders of magnitude of the two parts of the vector x, forces and displacements/velocities, and their residuals $R(x)$ may cause numerical issues as only the field with the larger magnitude is *seen* in the approximation of J^{-1}. For example, in case of a stiff solid structure, the surface pressure may be in the order of 10^5, whereas the structural deformation may be in the order of 10^{-3}. Additionally, the solution of the stiff structure might not change much between time steps, and therefore have a small residual value $R(x)$, while the residual of the fluid solver may still be several order of magnitude larger. Therefore, additional scaling of the sub-vectors of x and the respective residual components is required. We apply a pre-scaling to $V_k = \Lambda_k V_k$ and $r^k = \Lambda_k r^k$, where $\Lambda_k = \text{diag}(\lambda_{k,1} \dots \lambda_{k,1}\ \lambda_{k,2} \dots \lambda_{k,2})^T \in \mathbb{R}^{N \times N}$. The two pre-scaling weights $\lambda_{k,1}$ and $\lambda_{k,2}$ are selected to normalise the values of each sub-vector in r^k. Using the so-called residual-sum pre-scaling introduced in [24],

$$\lambda_{k,1} = \left(\sum_{j=1}^{k} \frac{\|S_2(x_2^j) - x_1^j\|_2}{\|R(x^j)\|_2} \right) \quad \text{and} \quad \lambda_{k,2} = \left(\sum_{j=1}^{k} \frac{\|S_1(x_1^j) - x_2^j\|_2}{\|R(x^j)\|_2} \right). \tag{15}$$

Uekermann [22] showed that this provided a suitable scaling for multiphysics simulations. In this pre-scaling, we use the norms of the fixed point equation residuals in each sub-field divided by the norm of the complete residual as scaling factors. This is then summed over all iterations, k, within one time step. The summation over all previous iterations in one time-step prevents a zig-zag convergence behaviour found in [22].

By updating the pre-scaling weights of Λ_k in each iteration, a complete QR-decomposition of V_k is required in every iteration instead of just a cheap update as all columns of V_k change. A complete QR-decomposition is required as each row in V_k is not scaled by the same value. This does not further affect the QR2 filter, but now a complete QR-decomposition in each iteration is also required for the QR1 filter.

2.4. Current Good Practice for IQN-ILS

Even though quasi-Newton methods for multiphysics coupling problems is advanced, tuning both convergence speed and runtime requires a careful understanding of the method in combination with the enhancements described above. Therefore, we describe some important aspects of IQN-ILS in combination with pre-scaling and filtering and the choice of other methodological parameters.

The pseudo-code of the IQN-ILS method with additional pre-scaling (Section 2.3.2) and filtering (Section 2.3.1) steps is shown in Algorithm 2. As the IQN-ILS method builds the approximation of J^{-1} exclusively from V_k and W_k, the problem dependent parameters η and ζ can have a large impact on the robustness and convergence as mentioned above. Whereas the general idea is to improve the approximation of J^{-1} by using more information than generated in a single time step, a large number of columns also has obvious negative impact: In addition to increasing the risk of rank deficiency of V_k (which can be resolved by filtering), retaining a large number of columns in W_k and V_k also leads to (i) larger memory consumption for matrix storage and (ii) larger computational effort in performing the QR-decomposition of V_k. A choice of between $\eta = 100$ and $\eta = 200$ previous iterations over the previous $\zeta = 10$ to $\zeta = 20$ time steps is a good choice if combined with a well-tuned filter. The filter *QR2* is preferred as it tends to delete older columns and only if they add little information to the QR-decomposition. The filter limit should be chosen such that some, but not too many columns are deleted. A good filter limit is typically between $\epsilon_f = 0.1$ and $\epsilon_f = 0.001$.

Algorithm 2 : IQN-ILS

```
initial value $x^0$
$\tilde{x}^0 = H(x^0)$ and $r^0 = \tilde{x}^0 - x^0$
$x^{k+1} = x^0 + \omega(\tilde{x}^0 - x^0)$
for $k = 1, 2, \ldots$ do
    $\tilde{x}^k = H\left(x^k\right)$ and $r^k = \tilde{x}^k - x^k$
    if converged then
        break
    end if
    $V_k = \left[\Delta r^k, \ldots, \Delta r^1\right]$, $\Delta r^k = r^k - r^{k-1}$
    $W_k = \left[\Delta \tilde{x}^k, \ldots, \Delta \tilde{x}^1\right]$, $\Delta \tilde{x}^k = \tilde{x}^k - \tilde{x}^{k-1}$
    Determine pre-scaling weights $\Lambda_k$
    Compute $\Lambda_k V_k = QR$
    Filter columns in $QR$
    solve $R\alpha = -Q^T \Lambda_k r^k$
    $x^{k+1} = \tilde{x}^k + W_k \alpha$
end for
```

3. Computational Improvements for the Quasi-Newton Method

The complex manner in which input parameters interact with each other makes it difficult to find the optimal input settings for a given problem, while still maintaining fast simulation runtimes. In the following section, we discuss two methods developed to improve the computational runtime while maintaining the robustness of the quasi-Newton methods.

3.1. Pre-Scaling Weight Monitoring

As the pre-scaling weights $\lambda_{k,1}$ and $\lambda_{k,2}$ change in each iteration, a complete QR-decomposition is performed in each iteration, adding a significant computational expense to the quasi-Newton update. We introduce a new pre-scaling weight monitoring method to freeze the pre-scaling weights after the first time step. In each iteration in the first time step, we recompute and apply the pre-scaling weights as usual. Starting from the second time step,, we only compute the *theoretical* new pre-scaling weight values Λ_k^* from the newest residual values using Equation (15). We update the actual pre-scaling weights $\Lambda_k \mapsto \Lambda_k^*$ if these *theoretical* weights change by more than one order of magnitude (i.e., if $\lambda_{k,i}^* > 10 \cdot \lambda_{k,i}$ or $\lambda_{k,i}^* < 0.1 \cdot \lambda_{k,i}$ for each solver i). This allows us to implement further methods to reduce the computational runtime for iterations where we keep pre-scaling weights constant:

1. We alter the QR-decomposition strategy for the QR1 filter, such that it only recomputes the QR-decomposition if pre-scaling weights change. Previously, the full re-computation was done in every iteration if pre-scaling was enabled.
2. We develop a new faster QR filter, QR3, that can mimic the behaviour of the QR2 filter. The previous QR2 inherently required to recompute the QR-decomposition in each iteration independent on whether pre-scaling was enabled. We describe the new filter below in Section 3.2).

3.2. Fast Alternative QR Filter

The current QR2 filtering technique relies on performing a complete QR-decomposition in every iteration. We typically assume that the number of DoF on the coupling interface, N, is much smaller than the number of DoF within the solvers. Therefore, the computational time for the QR-decomposition should be negligible. However, this may not always be the case as

1. the number η of columns in V_k can grow very large and the cost of inserting a column into QR has a computational complexity of $\mathcal{O}(\eta^3)$,
2. in volume coupling, the number of coupling DoF is equal to the number of all DoF in the domain, and is not negligible.

It is unnecessary for the QR2 filter step to perform a complete QR-decomposition if actually no column is deleted. In this case, a single column insertion step could have been performed. In this work, we introduce a new QR3 filter. A requirement for this filter is that the pre-scaling weights remain constant, and therefore the pre-scaling weight monitoring (Section 3.1) is required to use the QR3 filter. If the weights are updated during a coupling iteration, then a normal QR2 filter step is performed as the QR-decomposition is then rebuilt ($\Lambda_k V_k = QR$) in the process. Otherwise, the new filter computes these three steps:

1. the newest column of V_k is inserted into an existing QR decomposition (see Section 2.3.1),
2. a check is performed to tag any column that should be removed according to the same criteria as QR2,
3. only if any one column is tagged to be removed, then a normal QR2 filter step is performed instead, that is, a complete QR-decomposition is performed and columns are removed in this step.

In step 2, the check begins from the oldest column, $R_{:,\eta}$ (the subscript refers to the row and column number. $:,\eta$ refers to the right most column of R), and moves towards the previous newest column, $R_{:,2}$. We do not check if the first column $R_{:,1}$ should be removed as we want to keep the latest information. A column i is tagged for deletion if $R_{ii} < \epsilon_f \|V_{k,(:,i)}\|_2$. This criteria aims to mimic the behaviour of the QR2 filter by tending to find older columns to delete similar to the regular QR2 filter, but without reconstructing QR. If at least one column is tagged for deletion, then the QR2 filter is applied to rebuild QR starting from newest information, ensuring that a good quality of QR is maintained. Further runtime improvements could be found if this criterion is made stricter, for instance if at least two or three columns must be tagged for deletion before a QR2 filter step is performed. For simplicity, we restrict our tests, however, to only a single column. The *QR3* pseudo-code is shown in Algorithm 3.

Algorithm 3 : QR3 Filter

```
Add newest column $\overline{v} = (V_k)_{:,1}$ to $QR$
$filter \leftarrow$ false
for $i = \eta, ..., 2$ do                    ▷ Starts from oldest column and works forwards
    if $R_{ii} < \epsilon_f \|(V_k)_{:,i}\|_2$ then
        $filter \leftarrow$ true
        break
    end if
end for
if $filter$ then
    Compute QR2 Filter Step
end if
```

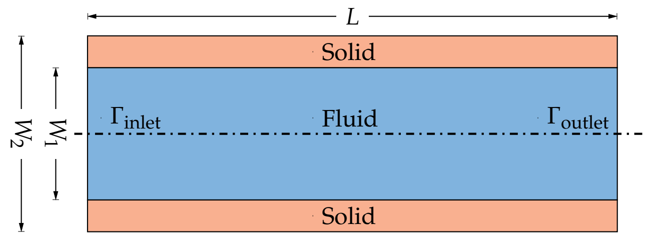
4. Numerical Setup

To test the effectiveness of our suggested enhancements, we introduce three fluid-structure interaction test scenarios in this section, followed by descriptions of the used software and hardware.

4.1. Test Cases

We selected three test scenarios that feature fluid and solid density values known to be challenging in terms of stability due to the added mass effect [25,26]: a 3D elastic tube scenario and a breaking dam scenario in 2D and 3D. Whereas the elastic tube scenario represents an outer elastic structure, the breaking dam scenarios feature elastic structures immersed in the fluid and involve free surface flow, such that we cover the main types of fluid-structure interaction scenarios. In addition, the breaking dam scenarios are dynamically changing, which challenges the reuse of past information of quasi-Newton methods. To ensure reproducibility of our results, the complete test setups are available under an open-source license at https://github.com/KyleDavisSA/IQN-test-cases accessed on 15 February 2022.

Elastic-Tube-3D. The Elastic-Tube-3D problem, proposed for instance in [27] and used in many other studies [8,13,20], is a simplified heamodynamic FSI test case. The test case geometry consists of a cylindrical tube with elastic walls and an inner fluid domain. A time-dependent pressure boundary condition is applied at the inlet and at the outlet of the fluid domain. At the inlet, a pressure of 1.3332 kPa is applied for 3 ms followed by 0 Pa for another 7 ms. At the outlet, the pressure is kept constant at 0 Pa. The simulation is run for 10^{-2} s, with a time step size of $dt = 10^{-4}$ s, for a total of 100 time steps. The fluid flow causes the solid domain to expand, and a pressure pulse travels through the tube. The domain geometry and material properties are shown in Figure 3.

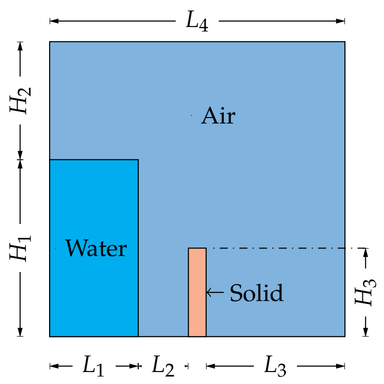


	Domain	
L	0.05	m
W_1	0.01	m
W_2	0.012	m
	Fluid	
ρ	1000	kg/m^3
ν	3×10^{-6}	m$^2 \times s^{-1}$
	Solid	
E	3×10^5	Pa
ρ	1200/120	kg/m^3

Figure 3. Domain geometry—not to scale—(**left**) and dimension and material parameters (**right**) of the Elastic-Tube-3D test case.

We compare two different densities for the structural solver to examine the added mass effect: (i) ρ = 1200 kg/m^3 and (ii) ρ = 120 kg/m^3, referred to as Elastic-Tube-3D-Heavy and Elastic-Tube-3D-Light, respectively. The structural solver has 11,735 elements in the domain and 1816 vertices on the coupling interface. The fluid domain contains 32,691 cells, with 1860 vertices on the coupling interface.

Breaking-Dam-2D. The Breaking-Dam-2D test case is a free surface problem, where a large body of water comes into contact with a flexible barrier [28,29]. This test case may pose problems for the quasi-Newton method: Firstly, the past information retained in the matrices V_k and W_k may not be entirely relevant once the water impacts the coupling interface and, thus, the character of the interaction between fluid and solid changes. Secondly, also the pre-scaling weight values may change dramatically at the moment of the impact. The domain and the material properties are shown in Figure 4.

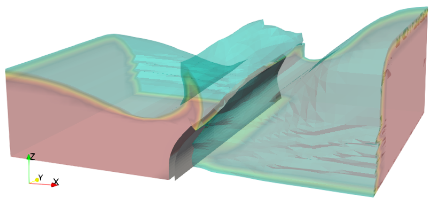


			Domain		
L_1	0.146	m	H_1	0.292	m
L_2	0.14	m	H_2	0.073	m
L_3	0.286	m	H_3	0.08	m
L_4	0.548	m			
	Fluid - Water			**Fluid - Air**	
ρ	1000	kg/m^3	ρ	1	kg/m^3
v	1×10^{-6}	$\text{m}^2 \times s^{-1}$	v	1×10^{-5}	$\text{m}^2 \times s^{-1}$
	Solid				
E	3×10^5	Pa			
ρ	2500	kg/m^3			

Figure 4. Domain geometry—not to scale—(**left**) and dimension and material parameters (**right**) of the Breaking-Dam-2D test case.

A no slip boundary condition is applied at the bottom, the left, and the right boundary, and a zero pressure outlet at the top. The test case was run for 1 s with a time step size of $dt = 0.005$ s, for a total of 200 time steps. The fluid domain contains 1382 cells in the domain, with 44 vertices on the interface, and the structural domain uses 325 quadratic finite elements in the domain, and 282 vertices on the coupling interface.

Breaking-Dam-3D. The Breaking-Dam-3D test case is a more complex test case inspired by the Breaking-Dam-2D example. A new, larger domain was created with larger bodies of water placed on either side of the wall, and a heavier solid wall was placed between the water columns. The water bodies are offset in the third dimension such that they hit the wall at opposite ends and at different times, resulting in a non-symmetrical movement of the dam wall (Figure 5 (left)). The solid domain is fixed only at the bottom and the sides are free to move in-plane. The test case was run for 0.75 s with a time step of $dt = 0.005$ s, for a total of 150 time steps. The domain geometry is shown in Figure 6, with the dimension and material properties given in Figure 5 (right). The fluid domain contains 25,712 cells in the domain, with 714 vertices on the interface. The solid domain is simulated using 319 linear elements with 387 vertices on the coupling interface.



			Domain		
L_1	2	m	H_2	2	m
L_2	0.95	m	H_3	1	m
L_3	1.95	m	H_4	0	m
L_4	1	m	H_5	4	m
L_5	6	m	H_6	1	m
H_1	2	m	H_7	4	m
	Fluid - Water			**Fluid - Air**	
ρ	1000	kg/m^3	ρ	1	kg/m^3
v	1×10^{-6}	$\text{m}^2 \times s^{-1}$	v	$1{\times}10^{-5}$	$\text{m}^2 \times s^{-1}$
	Solid				
E	1×10^7	Pa			
ρ	7850	kg/m^3			

Figure 5. Breaking-Dam-3D waters striking the flexible wall at 0.75 s (**left**) and dimensions and material parameters (**right**).

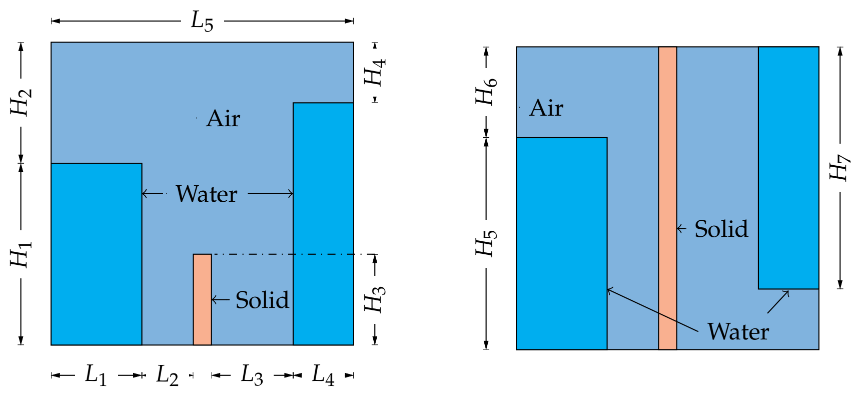


Figure 6. Domain geometry—not to scale—of the Breaking-Dam-3D test cases: front view (**left**), and top view (**right**).

4.2. *Quasi-Newton Configuration*

A few numerical parameters are the same for each test case. The following values are used unless stated otherwise:

- number of time steps reused: $\zeta = 10$ or $\zeta = 20$;
- maximal number of previous iterations: $\eta = 100$ or $\eta = 200$;
- convergence threshold for IQN-ILS: $\epsilon_{conv} = 10^{-3}$;
- maximum number of iterations allowed per time step before proceeding to the next time step, even if ϵ_{conv} is not reached: 30 or 50 (Breaking-Dam-3D);
- limit for QR2 or QR3 filter: $\epsilon_f = 0.001$, $\epsilon_f = 0.01$, or $\epsilon_f = 0.1$;
- type of pre-scaling: residual-sum pre-scaling as defined in Equation (15);
- initial under-relaxation value: $\omega = 0.1$.

A simulation run is denoted as diverged only if one of the physics solvers crashed, not if the convergence threshold was not reached in one or several time steps.

4.3. *Software*

We only use open-source software to test the quasi-Newton enhancements. We use the following versions:

- Fluid solver: https://www.openfoam.com/news/main-news/openfoam-v20-12, accessed on (OpenFOAM v2012) [30]; 15 February 2022
- Fluid solver adapter: https://github.com/precice/openfoam-adapter/releases/tag/v1.0.0, accessed on OpenFOAM-preCICE Adapter v1.0.0; 15 February 2022
- Solid solver: http://www.calculix.de/, accessed on CalculiX v2.17 [31]; 15 February 2022
- Solid solver adapter: https://github.com/precice/calculix-adapter/tree/5d42fb6160ede35926a59786ef8ae25dd71d7cdb, accessed on CalculiX-preCICE Adapter, commit 5d42fb6, 15 February 2022.

All quasi-Newton methods described in Sections 2 and 3 have been implemented in the coupling library preCICE [4,7]. We use two different versions of preCICE:

- https://github.com/precice/precice/releases/tag/v2.3.0 accessed on 15 February 2022, as baseline without the enhancements presented in Section 3
- https://github.com/precice/precice/tree/3fb3d8d465e45e1eadba766a8ce5f1f96c138b20 accessed on 15 February 2022, for the enhancements presented in Section 3.

4.4. *Hardware*

All simulations are run on a single core of a Lenovo T480, with an Intel Core i5–8250U CPU, 1.60 GHz $\times$ 4, and 16 GB main memory.

5. Results and Discussion

In this section, we present the results and discussion for the pre-scaling weight monitoring and the new QR3 filtering procedure.

5.1. Pre-Scaling Weight Monitoring

We introduced pre-scaling weight monitoring in order to reduce the number of weight updates throughout the simulation and, as a consequence, to be able to reduce the computational cost of QR-decompositions and filtering. Therefore, we examine the impact of different variants of pre-scaling weight updating on both, the number of quasi-Newton iterations and on the computational cost for QR-decompositions and filtering.

Table 1 shows the impact of different versions of pre-scaling on the number of quasi-Newton iterations. First, we provide a baseline set of results where we update the pre-scaling weights in each iteration. Second, we freeze the pre-scaling weights after the first time step. Finally, we use the pre-scaling weight monitoring.

Table 1. Comparison of the average number of quasi-Newton iterations per time step for IQN-ILS with QR2 filter ($\epsilon_f = 0.01$) for (i) pre-scaling update in every quasi-Newton iteration (column "baseline"), (ii) pre-scaling with freezing of weights after the first time step (column "freeze"), (iii) the new pre-scaling weight monitoring approach (column "monitoring"). For each test case, the first row uses a maximum of $\eta = 100$ iterations from previous $\zeta = 10$ time steps, and the second row $\eta = 200$ and $\zeta = 20$. Braces indicate how many time steps did not converge within 30 iterations before moving to the next time step.

Test Case	ζ	η	Baseline	Freeze	Monitoring
Elastic-Tube-3D Heavy	10	100	4.51	4.45	4.48
Elastic-Tube-3D Heavy	20	200	3.84	3.96	3.94
Elastic-Tube-3D Light	10	100	7.23	7.30	7.16
Elastic-Tube-3D Light	20	200	5.84	6.00	5.83
Breaking-Dam-2D	10	100	7.08 (2)	8.59 (8)	7.9 (3)
Breaking-Dam-2D	20	200	8.06 (6)	9.16 (10)	8.14 (2)
Breaking-Dam-3D	10	100	4.34	4.77	4.33
Breaking-Dam-3D	20	200	4.3	5.19	4.15

From the comparison of quasi-Newton iteration counts in Table 1, we observe that, in general, our new pre-scaling weight monitoring approach does not lead to an increase of the number of iterations compared to the more expensive baseline approach, where weights are updated in every iteration. All test cases apart from the Breaking-Dam-2D case offer comparable results in terms of iteration counts. For the Breaking-Dam-2D case, a slight increase can be observed for our weight monitoring approach. However, overall, the results show that the weight monitoring is effective in the sense that it detects when updates are necessary.

On the other hand, comparing our new approach to simulation runs where we freeze the pre-scaling weights after the first time step shows that freezing the weights substantially increases the number of iterations. It yields comparable results only for both Elastic-Tube-3D cases. There is a significant increase in the number of iterations for both Breaking-Dam cases, which shows that simply freezing the weights is sub-optimal if scenarios undergo sudden changes (fluid hitting the obstacle in the Breaking-Dam case). We conclude, that the weight monitoring is not only sufficient, but also necessary to ensure fast quasi-Newton convergence in general.

To further analyse the impact of the pre-scaling weight monitoring on the overall computational cost, we present more detailed results for the fluid and the solid domain in Table 2, in particular including the number of weight updates, the most important factor for computational cost as each update requires a complete re-computation of the QR-decomposition of V_k. Here, we use the pre-scaling weight monitoring with the new

QR3 filter. The combined impact of this new filter and the pre-scaling weight monitoring is presented in Section 5.2.

The more detailed analysis of these tests in Table 2 leads to further insights: (i) The large difference between the fluid and solid pre-scaling weights is immediately apparent. This large difference necessitates using pre-scaling in the first place. Completely removing the pre-scaling results in either a large increase in coupling iterations, or divergence of the solvers, for all test cases. For brevity, we do not explicitly list these results. (ii) In addition, we observe a large difference between the minimum and the maximum value of the weights for each of the fields over the entire simulation—fluid and solid. This indicates that updating the pre-scaling weights is necessary to ensure suitable scaling.

Table 2. Performance details of pre-scaling weight monitoring with QR3 filter ($\epsilon_f = 0.01$). For each test case, the first row uses a maximum of $\eta = 100$ iterations from previous $\zeta = 10$ time steps, and the second row $\eta = 200$ and $\zeta = 20$. We show the total number of quasi-Newton iterations for the entire simulation with the average number of quasi-Newton iterations per time step in braces (column "Its"), the numbers of updates of the pre-scaling weights during both the first time step (column "Upd 1") and all other time steps (column "Upd 2-"), as well as the range of weights $\lambda_{f/s}$ for both solvers.

Test Case	ζ	η	Its	Upd 1	Upd 2-	λ_s	λ_f
Elastic-Tube-3D-Heavy	10	100	448 (4.48)	14	1	17.38–3155.7	0.071–1.0
Elastic-Tube-3D-Heavy	20	200	394 (3.94)	14	1	17.38–3155.7	0.071–1.0
Elastic-Tube-3D-Light	10	100	716 (7.16)	22	6	4.68–2365.8	0.045–1.0
Elastic-Tube-3D-Light	20	200	583 (5.83)	22	7	2.13–2401.5	0.045–1.0
Breaking-Dam-2D	10	100	1525 (7.63)	4	54	0.98–1018.5	0.09–1.0
Breaking-Dam-2D	20	200	1651 (8.23)	4	75	0.98–621.3	0.09–1.0
Breaking-Dam-3D	10	100	649 (4.33)	10	11	9.67×10^4–3.23×10^7	0.1–1.0
Breaking-Dam-3D	20	200	623 (4.15)	10	6	2.19×10^5–3.23×10^7	0.1–1.0

During the first time step, the pre-scaling weight monitoring updates the weights in each iteration. Therefore, a large number of weight updates is observed for the first time step in Table 2 (column "Upd 1") compared to the rest of the simulation (column "Upd 2-"). For all test cases, the number of updates after the first time step is low compared to the total number of iterations, with the Elastic-Tube-3D-Light cases requiring only six and seven updates after the first time step, respectively. The Breaking-Dam-2D scenario requires 54 and 75 updates after the first time step. However, this is a low number compared to the total of 1525 and 1651 iterations for the entire simulation.

Comparing the amount of retained iterations η over a certain window of time steps ζ, we observe that the number of pre-scaling weight updates as well as the range of weight values do not appear to be very sensitive to these parameters. This was expected since converged solutions at the end of each time step should be rather independent of these parameters.

The range of pre-scaling weight values is also interesting. The Breaking-Dam-3D case features very large scaling values for the solid solver. This indicates that the solid solver residuals are far smaller than the fluid solver's residuals. For this test case, the structure is a heavy wall and "slow" to move. This could account for a high stability in the solid solver itself, but necessitates the use of pre-scaling. A change in the pre-scaling magnitudes is observed for the Elastic-Tube-3D scenarios, with slightly larger solid solver pre-scaling weight values for the Elastic-Tube-3D-Heavy scenario with a higher structure density, once again indicating smaller residual values compared to the fluid solver.

Summarising, our results show that the pre-scaling weights do not change considerably between successive iterations majority of the time, and that the factor of 10 used to determine when the pre-scaling weights is updated is a suitable choice. However, the weights can change significantly over time, and adjusting the weights is necessary.

5.2. QR3 Filter

We introduced the QR3 filter in order to reduce the number of complete QR-decompositions performed when filtering columns from QR in the quasi-Newton update. The new QR3 filter mimics the previous QR2 filter method in the way it detects which columns are to be deleted in V_k and W_k. However, it does not require a complete QR-decomposition in each quasi-Newton iteration, but only cheap QR-updates as long as (i) pre-scaling weights are unchanged and (ii) no column is tagged to be deleted. In the experiments in this section, we analyse the impact of the new filter in combination with pre-scaling weight monitoring on both the number of required quasi-Newton iterations and on the computational runtime. We compare the number of quasi-Newton iterations in Table 3 and the number of deleted columns in Table 4 for the QR2 filter while computing new pre-scaling weights in each iteration, and the new QR3 filter with pre-scaling weight monitoring.

Table 3. Comparison of the average number of quasi-Newton iterations per time step for (i) pre-scaling update in every quasi-Newton iteration with the QR2 filter (column "QR2"), and (ii) the new pre-scaling weight monitoring approach with the new QR3 filter (column "QR3"). Three different filter limits $\epsilon_f = 0.001$, $\epsilon_f = 0.01$, and $\epsilon_f = 0.1$ are compared. Values in brackets indicate how many time steps did not converge within 30 iterations before moving to the next time step.

				QR2			**QR3**	
Test Case	ζ	η	$\epsilon_f = 0.001$	**0.01**	**0.1**	$\epsilon_f = 0.001$	**0.01**	**0.1**
Elastic-Tube-3D-Heavy	10	100	4.26	4.51	5.12	4.59	4.48	5.24
Elastic-Tube-3D-Heavy	20	200	4.09	3.84	4.91	4.05	3.94	3.92
Elastic-Tube-3D-Light	10	100	7.27	7.23	8.83	7.18	7.16	8.69
Elastic-Tube-3D-Light	20	200	5.78	5.84	7.88	5.83	5.83	7.67
Breaking-Dam-2D	10	100	div	7.08 (2)	5.87 (3)	12.5 (25)	7.63 (3)	5.76 (2)
Breaking-Dam-2D	20	200	12.12 (19)	8.10 (6)	6.11 (2)	div	8.26 (2)	5.74 (2)
Breaking-Dam-3D	10	100	4.45	4.34	4.21	4.8	4.33	4.31
Breaking-Dam-3D	20	200	4.51	4.30	4.15	4.51	4.15	4.33

Table 4. Average number of columns deleted per time step for (i) pre-scaling update in every quasi-Newton iteration with the QR2 filter (column "QR2"), and (ii) the new pre-scaling weight monitoring approach with the new QR3 filter (column "QR3"). Three different filter limits $\epsilon_f = 0.001$, $\epsilon_f = 0.01$, and $\epsilon_f = 0.1$ are compared.

				QR2			**QR3**	
Test Case	ζ	η	$\epsilon_f = 0.001$	**0.01**	**0.1**	$\epsilon_f = 0.001$	**0.01**	**0.1**
Elastic-Tube-3D-Heavy	10	100	0.01	0.1	2.01	0.01	0.03	1.61
Elastic-Tube-3D-Heavy	20	200	0.01	0.04	2.19	0.01	0.03	0.31
Elastic-Tube-3D-Light	10	100	0.01	0.04	2.67	0.01	0.01	2.88
Elastic-Tube-3D-Light	20	200	0.01	0.03	2.8	0.01	0.03	2.65
Breaking-Dam-2D	10	100	div	3.45	3.95	4.42	3.82	3.8
Breaking-Dam-2D	20	200	8.25	5.81	4.64	div	5.81	4.19
Breaking-Dam-3D	10	100	1.99	2.06	2.19	2.14	1.88	2.17
Breaking-Dam-3D	20	200	2.03	2.27	2.45	2.26	2.03	2.38

The number of quasi-Newton iterations as presented in Table 3 does not increase when using the new QR3 filter instead of the QR2 filter. It even decreases for 13 of the 24 simulations performed. The Breaking-Dam-2D case diverges for $\epsilon_f = 0.001$ for both filters. This is in-line with previous observations, that for scenarios that are prone to linear dependencies in V_k, the filter limit cannot be chosen too small [10,22]. Changing from the QR2 to the QR3 filter is, thus, not the source of this issue.

Table 4 shows that also the number of deleted columns is comparable for both filters. The Elastic-Tube-3D-Heavy test case has only one and three columns deleted for $\epsilon_f = 0.001$ and $\epsilon_f = 0.01$, respectively. The Breaking-Dam test cases have more columns deleted, approximately half of all iterations. Both filters are able to address this with similar efficiency.

For $\epsilon_f = 0.001$, the filtering criterion tends to remove fewer columns. This seems to be somewhat contradicted by the Breaking-Dam-3D case with the QR3 filter, where the number of deleted columns is larger for $\epsilon_f = 0.001$. However, the total number of iterations for $\epsilon_f = 0.001$ is also larger than for $\epsilon_f = 0.01$. Therefore, more QR filter checks are performed and more columns are deleted over the entire simulation runtime. The same happens for the Breaking-Dam-2D case, where the number of removed columns for $\epsilon_f = 0.01$ is larger than for $\epsilon_f = 0.1$.

The runtime of the QR filter (filtering of columns including QR decomposition time) as a percentage of the total simulation runtime is shown in Table 5. The total simulation runtime includes the total runtime of the solvers and preCICE including initialisation.

Table 5. QR decomposition and filtering time as a percentage of the total simulation runtime for (i) pre-scaling update in every quasi-Newton iteration with the QR2 filter (QR2), and (ii) the new pre-scaling weight monitoring approach with the new QR3 filter (QR3). Three filter limits $\epsilon_f = 0.001$, $\epsilon_f = 0.01$ and $\epsilon_f = 0.1$ were tested. The test cases used a maximum of $\eta = 100$ iterations from previous $\zeta = 10$ time steps, and $\eta = 200$ and $\zeta = 20$.

				QR2			**QR3**	
Test Case	ζ	η	ϵ_f **= 0.001**	**0.01**	**0.1**	ϵ_f **= 0.001**	**0.01**	**0.1**
Elastic-Tube-3D-Heavy	10	100	1.99%	2.48%	1.51%	0.06%	0.06%	0.62%
Elastic-Tube-3D-Heavy	20	200	6.67%	4.9%	3.18%	0.08%	0.09%	0.36%
Elastic-Tube-3D-Light	10	100	7.41%	7.23%	6.60%	0.15%	0.15%	2.42%
Elastic-Tube-3D-Light	20	200	12.99%	13.12%	13.20%	0.38%	0.43%	5.04%
Breaking-Dam-2D	10	100	div	4.69%	1.27%	3.21%	2.52%	0.85%
Breaking-Dam-2D	20	200	27.98%	9.72%	2.33%	div	5.75%	1.65%
Breaking-Dam-3D	10	100	0.61%	0.59%	0.54%	0.08%	0.03%	0.04%
Breaking-Dam-3D	20	200	1.75%	1.57%	1.44%	0.23%	0.09%	0.07%

Comparing the relative runtimes shown in Table 5, we see that, for every test case, the runtime of the new QR3 filter is significantly smaller than the runtime of the QR2 filter. More noticeable improvements occur for $\zeta = 20$ and $\eta = 200$ than for $\zeta = 10$ and $\eta = 100$, which is expected since the larger number of columns in V_k increases the cost of each complete QR-decomposition. The filtering accounts for a large percentage of the simulation runtime for the Breaking-Dam-2D case, as the solver meshes are rather small. Especially for $\zeta = 20$ and $\eta = 200$ and the QR2 filter with limit $\epsilon_f = 0.001$, the QR-decompositions are relatively expensive. The Elastic-Tube-3D-Light scenario spends up to 13.20% of the simulation runtime filtering the QR2 filter with $\epsilon_f = 0.1$. The largest improvement in runtime performance is approximately 12.5% of the simulation runtime for the Elastic-Tube-3D-Light with $\zeta = 20$ and $\eta = 200$.

Also in terms of runtime and runtime gains, we observe large differences between Breaking-Dam-2D and Breaking-Dam-3D scenarios, which can, however, be easily explained as the 2D problem has a much smaller domain in terms of size and number of elements in the domain. Each fluid solver call is relatively "cheap" computationally, and therefore the relative cost of the QR decomposition increases.

Note that these runtime comparisons represent the overall gain of both improvements, pre-scaling weight monitoring and the QR3 filter as the QR2 filter is not able to exploit the advantage of pre-scaling weight monitoring, that is, it always re-computes the complete QR-decomposition of V_k independent of whether weight updates are required or not.

6. Conclusions

We provide an overview of current quasi-Newton acceleration methods commonly used for partitioned fluid-structure interaction. We also discuss numerical techniques that improve the computational efficiency and enhance convergence of quasi-Newton acceleration. From this, two new methods are introduced to further improve the computational efficiency of the acceleration: pre-scaling weight monitoring and a faster QR filtering procedure. The combination of both methods allows us to significantly reduce the necessary number of computationally expensive QR-decompositions. Instead, we can simply update existing QR decompositions in most acceleration steps—a rather cheap operation. We study the effectiveness of these new methods with three common fluid-structure interaction test cases, each offering a unique coupling difficulty. The new methods reduce the runtime of the QR filter significantly, while not decreasing the convergence speed. A small, but already significant speed up for complete fluid-structure interaction simulations is observable. This is true despite the fact that there are drastically fewer degrees of freedom at the coupling interface compared to the solver domains for such surface-coupled problems. For volume-coupled problems, we expect an even higher impact of the newly introduced methods.

Author Contributions: Conceptualization, K.D.; sata curation, K.D.; formal analysis, K.D.; funding acquisition, M.S. and B.U.; investigation, K.D.; methodology, K.D.; project administration, M.S.; software, K.D. and B.U.; supervision, M.S. and B.U.; validation, K.D.; visualization, K.D.; writing—original draft, K.D.; writing—review and editing, K.D., M.S. and B.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number ME 2067/2-1 and under Germany's Excellence Strategy—EXC 2075—390740016.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found at: preCICE Repository: https://github.com/precice/precice 15 February 2022, preCICE Tutorials Repository: https://github.com/precice/tutorials 15 February 2022, Test cases and Configuration Files: https://github.com/KyleDavisSA/IQN-test-cases 15 February 2022.

Acknowledgments: We would like to thank the preCICE development team, and all those who have helped develop preCICE. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech).

Conflicts of Interest: The funders had no role in the design of the study, in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Grognuz, J. A New Heart Valve Replacement Procedure Modeled with Multiphysics Simulation Could Eliminate the Need for Open-Heart Surgery. Available online: https://www.enginsoft.com/expertise/a-new-heart-valve-replacement-procedure.html (accessed on 2 February 2021).
2. Jain, R.K.; Saha, P. *Fluid-Structure Interaction Simulations Prove Ability of Solar Artifacts to Withstand Wind Gusts*; ANSYS Inc.: Canonsburg, PA, USA, 2021. Available online: https://www.ansys.com/content/dam/product/3d-design/aim/csir-cmeri-cs.pdf (accessed on 15 February 2022).
3. Schmidt, P.; Jaust, A.; Steeb, H.; Schulte, M. Simulation of flow in deformable fractures using a quasi-Newton based partitioned coupling approach. *Comput. Geosci.* **2022**, *26*, 381–400. [CrossRef]
4. Bungartz, H.J.; Lindner, F.; Gatzhammer, B.; Mehl, M.; Scheufele, K.; Shukaev, A.; Uekermann, B. preCICE—A fully parallel library for multi-physics surface coupling. *Comput. Fluids* **2016**, *141*, 250–258. [CrossRef]
5. Slattery, S.; Wilson, P.P.H.; Pawlowski, R.P. The data transfer kit: A geometric rendezvous-based tool for multiphysics data transfer. In Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013), Sun Valley, ID, USA, 5–9 May 2013.
6. Duchaine, F.; Jauré, S.; Poitou, D.; Quémerais, E.; Staffelbach, G.; Morel, T.; Gicquel, L. Analysis of high performance conjugate heat transfer with the OpenPALM coupler. *Comput. Sci. Discov.* **2015**, *8*, 015003. [CrossRef]
7. Chourdakis, G.; Davis, K.; Rodenberg, B.; Schulte, M.; Simonis, F.; Uekermann, B. preCICE V2: A Sustainable and User-Friendly Coupling Library. *arXiv* **2021**, arXiv:2109.14470.
8. Degroote, J.; Bathe, K.J.; Vierendeels, J. Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction. *Comput. Struct.* **2009**, *87*, 793–801. [CrossRef]

9. Bogaers, A.E.J.; Kok, S.; Reddy, B.D.; Franz, T. Quasi-Newton methods for implicit black-box FSI coupling. *Comput. Methods Appl. Mech. Eng.* **2014**, *279*, 113–132. [CrossRef]
10. Haelterman, R.; Bogaers, A.; Uekermann, B.; Scheufele, K.; Mehl, M. Improving the performance of the partitioned QN-ILS procedure for fluid-structure interaction problems: Filtering. *Comput. Struct.* **2016**, *171*, 9–17. [CrossRef]
11. Mehl, M.; Uekermann, B.; Bijl, H.; Blom, D.; Gatzhammer, B.; van Zuijlen, A. Parallel coupling numerics for partitioned fluid–structure interaction simulations. *Comput. Math. Appl.* **2016**, *71*, 869–891. [CrossRef]
12. Scheufele, K.; Mehl, M. Robust multisecant Quasi-Newton variants for parallel fluid-structure simulations and other multiphysics applications. *SIAM J. Sci. Comput.* **2016**, *39*, 404–433. [CrossRef]
13. Spenke, T.; Hosters, N.; 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. [CrossRef]
14. Anderson, D.G. Iterative procedures for nonlinear integral equations. *J. ACM* **1965**, *12*, 547–560. [CrossRef]
15. Miller, K. Nonlinear krylov and moving nodes in the method of lines. *J. Comput. Appl. Math.* **2005**, *183*, 275–287. [CrossRef]
16. Ni, P. Anderson Acceleration of Fixed-Point Iteration with Applications to Electronic Structure Computations. Ph.D. Thesis, Worcester Polytechnic Institute, Worcester, MA, USA, 13 November 2009. Available online: https://www.semanticscholar.org/paper/Anderson-Acceleration-of-Fixed-point-Iteration-with-Ni/8ca4703c5ec5c4580950a9c5c806604a595db3cb (accessed on accessed on 15 February 2022).
17. Oosterlee, C.W.; Washio, T. Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows. *SIAM J. Sci. Comput.* **2000**, *21*, 1670–1690. [CrossRef]
18. Risseeuw, D. Fluid Structure Interaction Modelling of Flapping Wings. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 2019.
19. Bungartz, H.J.; Lindner, F.; Mehl, M.; Uekermann, B. A plug-and-play coupling approach for parallel multi-field simulations. *Comput. Mech.* **2015**, *55*, 1119–1129. [CrossRef]
20. Lindner, F.; Mehl, M.; Scheufele, K.; Uekermann, B. A Comparison of various Quasi-Newton Schemes for Partitioned Fluid-Structure Interaction. In Proceedings of the VI International Conference on Computational Methods for Coupled Problems in Science and Engineering, Venice, Italy, 18–20 May 2015; pp. 477–488.
21. Scheufele, K. Coupling Schemes and Inexact Newton for Multi-Physics and Coupled Optimization Problems. Ph.D. Thesis, University of Stuttgart, Stuttgart, Germany, 2019.
22. Uekermann, B. Partitioned Fluid-Structure Interaction on Massively Parallel Systems. Ph.D. Thesis, Technical University of Munich, Munich, Germany, 2016.
23. Daniel, J.W.; Gragg, W.B.; Kaufman, L.; Stewart, G. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Math. Comput.* **1976**, *30*, 772–795. [CrossRef]
24. Marks, L.; Luke, D. Robust mixing for ab initio quantum mechanical calculations. *Phys. Rev. B* **2008**, *78*, 075114. [CrossRef]
25. Förster, C.; Wall, W.A.; Ramm, E. Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. *Comput. Methods Appl. Mech. Eng.* **2007**, *196*, 1278–1293. [CrossRef]
26. Van Brummelen, E.H. Added Mass Effects of Compressible and Incompressible Flows in Fluid-Structure Interaction. *J. Appl. Mech.* **2009**, *76*, 021206. [CrossRef]
27. Gerbeau, J.F.; Vidrascu, M. *A Quasi-Newton Algorithm Based on a Reduced Model for Fluid-Structure Interaction Problems in Blood Flows*; [Research Report] RR-4691, INRIA; Cambridge University Press: Cambridge, UK, 2003.
28. Bogaers, A.E.J.; Kok, S.; Reddy, B.D.; Franz, T. An evaluation of quasi-Newton methods for application to FSI problems involving free surface flow and solid body contact. *Comput. Struct.* **2016**, *173*, 71–83. [CrossRef]
29. Walhorn, E.; Kölke, A.; Hübner, B.; Dinkler, D. Fluid–structure coupling within a monolithic model involving free surface flows. *Comput. Struct.* **2005**, *83*, 2100–2111. [CrossRef]
30. Weller, H.G.; Tabor, G.; Jasak, H.; Fureby, C. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Comput. Phys.* **1998**, *12*, 620–631. [CrossRef]
31. Dhondt, G. *The Finite Element Method for Three-Dimensional Thermomechanical Applications*; John Wiley and Sons: Hoboken, NJ, USA, 2004.