Structural and Multidisciplinary Optimization manuscript No.
(will be inserted by the editor)

Partitioned solution of an unsteady adjoint for strongly coupled fluid-structure interactions and application to parameter identification of a one-dimensional problem

Joris Degroote · Majid Hojjat · Electra Stavropoulou · Roland Wüchner · Kai-Uwe Bletzinger

Received: date / Accepted: date

J. Degroote
Ghent University
Department of Flow, Heat and Combustion Mechanics
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
Tel: +32 9 264 95 22
Fax: +32 9 264 35 86
E-mail: Joris.Degroote@UGent.be

M. Hojjat
Technische Universität München
Chair of Structural Analysis
Arcisstrasse 21, 80333 Munich, Germany
Tel: +49 89 289 22152
Fax: +49 89 289 22421
E-mail: hojjat@tum.de

E. Stavropoulou
Technische Universität München
Chair of Structural Analysis
Arcisstrasse 21, 80333 Munich, Germany
Tel: +49 89 289 22152
Fax: +49 89 289 22421
E-mail: stavropoulou@tum.de

R. Wüchner
Technische Universität München
Chair of Structural Analysis
Arcisstrasse 21, 80333 Munich, Germany
Tel: +49 89 289 22423
Fax: +49 89 289 22421
E-mail: wuechner@tum.de

K.-U. Bletzinger
Technische Universität München
Chair of Structural Analysis
Arcisstrasse 21, 80333 Munich, Germany
Tel: +49 89 289 22422
Fax: +49 89 289 22421
E-mail: kub@tum.de

Abstract Unsteady fluid-structure interaction (FSI) simulations are generally time-consuming. Gradient-based methods are preferred to minimise the computational cost of parameter identification studies (and more in general optimisation) with a high number of parameters. However, calculating the cost function's gradient using finite differences becomes prohibitively expensive for a high number of parameters. Therefore, the adjoint equations of the unsteady FSI problem are solved to obtain this gradient at a cost almost independent of the number of parameters.

Here, both the forward and the adjoint problems are solved in a partitioned way, which means that the flow equations and the structural equations are solved separately. The application of interest is the identification of the arterial wall's stiffness by comparing the motion of the arterial wall with a reference, possibly obtained from non-invasive imaging. Due to the strong interaction between the fluid and the structure, quasi-Newton coupling iterations are applied to stabilise the partitioned solution of both the forward and the adjoint problem.

Keywords Adjoint · Coupled · Partitioned · Fluid-structure interaction · Quasi-Newton

1 Introduction

In this work, the stiffness parameters of an elastic tube through which an incompressible fluid flows are identified. The tube model is a one-dimensional generalised string model and the stiffness of each tube segment is determined by a separate parameter, corresponding with a spatially distributed parameter. This tube is a simplified fluid-structure interaction model for the blood flow in a large artery [1]. Previous research has demonstrated that this model results in a finite wave propagation speed due to the fluid-structure interaction, despite the incompressible fluid [2]. Nevertheless, this model is not suitable to calculate wall shear stress due to the abscence of viscosity.

The goal is to adjust the stiffness parameters of this model so that the displacement of the tube wall as a function of time matches the displacement data from a non-invasive measurement. This stiffness is considered as an indicator for arterial diseases, but it cannot be measured directly in a non-invasive way. Hence, parameter identification using computer simulations is relevant.

A typical simulation of this or any other model yields an amount of data for a given value of the model parameters. Such a simulation solves a so-called forward (or direct) problem. The corresponding inverse problem is to find parameter values such that the data resulting from a forward simulation with these parameter values agrees as closely as possible with some measurement (or target) [3]. Parameter identification techniques to solve these inverse problems can generally be reformulated as a minimisation problem. In that case, a cost function is defined as the difference between the simulation data and the measurement. The parameter values which minimise this cost function are

then determined using an optimisation algorithm or filtering [4]. The optimisation approach is selected here, so that the developed techniques are applicable not only to parameter identification, but also to other optimisation problems in general.

Optimisation algorithms can be classified into methods that only evaluate the cost function value and those which also use the gradient (first derivatives of the cost function with respect to the parameters) or even the Hessian (second derivatives of the cost function with respect to the parameters). Methods that only evaluate the cost function include genetic algorithms [5], swarm intelligence [6], simulated annealing [7] and surrogate-based methods with adaptive sampling [8], among others. These methods can find the global optimum of non-differentiable functions but the number of cost function evaluations generally increases drastically as the number of parameters increases. As a result, the computational cost becomes prohibitive, especially when simulation of the forward problem is time-consuming [9].

By contrast, gradient-based optimisation algorithms can solve optimisation problems with a high number of parameters using a relatively limited number of cost function evaluations. Steepest descent, conjugate gradient and quasi-Newton methods belong to this class. However, these algorithms tend to converge to local optima. Moreover, obtaining the gradient is in most cases not trivial. Because the number of parameters is high in the application of interest, gradient-based optimisation is applied in this work [9]. Both the gradient and the exact Hessian of the cost function are used by another class of optimisation algorithms, including Newton's method.

Several possibilities exist for the gradient calculation. It can be calculated using finite differences, the direct method and the adjoint method [10,11]. These techniques will be described in detail later in this work. The adjoint method is chosen because its computational cost does not significantly depend on the number of parameters, which is advantageous for a high number of parameters. The difference between the discrete and continuous adjoint formulation is that the discretisation occurs respectively before and after the derivation of the adjoint equations. Here, the discrete adjoint formulation is selected to obtain the exact gradient of the discrete equations.

The fluid-structure interaction in the model can be dealt with in two ways. In the monolithic approach, the flow equations and structural equations are solved simultaneously [12–14]. Conversely, the flow equations and structural equations are solved separately in the partitioned approach. To obtain software modularity, the partitioned approach is selected for this research. In weakly (or explicitly, staggered) coupled partitioned simulations, the flow equations and the structural equations are solved once (or a limited, fixed number of times) per time step [15,16]. As a result, the equilibrium of force and velocity on the fluid-structure interface is not guaranteed. In strongly (or implicitly) coupled partitioned simulations, coupling iterations are performed between the flow equations and the structural equations in each time step. Consequently, the equilibrium conditions on the interface are satisfied (up to an error proportional to the convergence tolerance of the coupling iterations). The weakly

coupled methods are suitable for compressible aeroelastic simulations, while strong coupling should be used for simulations with incompressible fluids [17].

Several strongly coupled partitioned techniques treat the flow solver and the structural solver as black boxes, i.e. programmes which calculate an output for a given input, but whose internal algorithms can neither be accessed nor modified. These methods include Gauss-Seidel iterations, Gauss-Seidel iterations with Aitken relaxation [18–20], interface GMRES [21] and interface quasi-Newton (IQN-ILS) iterations [22,23]. However, Gauss-Seidel iterations are often unstable if the ratio of the fluid density to the structure density is high, among other reasons [24,25]. In comparisons consisting of several test cases, IQN-ILS requires fewer coupling iterations per time step than Aitken relaxation or Interface GMRES [22], and the computational cost ratio of a partitioned simulation with IQN-ILS to a monolithic simulation ranges from 0.56 to 3.16 [26]. Therefore, IQN-ILS is selected as coupling algorithm.

In this work, the adjoint equations for an unsteady fluid-structure interaction problem are derived for a spatially distributed parameter. Both the forward and the adjoint equations are solved in a partitioned way using the IQN-ILS coupling algorithm. A gradient for a spatially distributed parameter calculated using adjoint equations has already been combined with optimisation of fluid-structure interaction by several authors. For example, it has been applied for the aerostructural shape optimisation of an aeroplane [27, 28] and its wings [29–32]. However, these are steady problems which can be solved using Gauss-Seidel iterations, while Gauss-Seidel iterations are unstable for the unsteady model at hand. In [33,34], the adjoint of the steady Euler equations is calculated and the resulting gradient is inserted in an unsteady linear aeroelastic model with four equations to derive a controller for flutter reduction.

Unsteady adjoint solvers have been developed for the optimisation of fluids and structures alone. For example, an adjoint solver of the unsteady Euler equations has been used for the optimisation of blast mitigation devices [9]. Also adjoint solvers of the unsteady Navier-Stokes equations exist [35], even for moving and deforming grids [36]. Checkpointing is often applied to reduce the memory requirements of unsteady adjoint simulations [37–39], but this is not necessary in this case.

In the biomedical field, a number of authors previously applied parameter identification to determine the stiffness of an artery. In [40], three parameters are identified using continuous adjoint equations, while in [41] one stiffness parameter is calculated using discrete adjoint equations so that a monolithic one-dimensional model corresponds with a three-dimensional model. In [4], five stiffness parameters of a three-dimensional model are identified using a reduced-order unscented Kalman filter (UKF). This approach requires as many simulations as there are parameters plus one, which run simultaneously.

The main novelty of this work is thus the partitioned solution of the adjoint equations for an unsteady fluid-structure interaction problem with a spatially distributed parameter. Both the forward and adjoint equations are solved using quasi-Newton coupling since Gauss-Seidel iterations are unstable, as will

be demonstrated by the numerical results. The quasi-Newton coupling algorithm for FSI problems has already been published before [22]. Within this contribution, this algorithm is derived again to obtain a consistent notation for physical state analysis and coupled adjoint computation. Furthermore, it is demonstrated how its advantageous properties can be exploited for the calculation of unsteady coupled sensitivities. Insights from the field of partitioned simulations are thereby transferred to the fields of sensitivity analysis and adjoint equations. The model itself is intentionally kept simple and should be considered as a demonstration example.

The remainder of this article is organised as follows. The equations and solution procedure are described in Section 2 for the forward problem and in Section 3 for the adjoint problem. Section 4 explains the optimisation procedure, followed by the details of the implementation in Section 5. Finally, the results and conclusions are presented in Section 6 and Section 7, respectively.

2 Model

2.1 Continuous equations

The model for the parameter identification of the blood flow in an artery is the unsteady flow of an incompressible, inviscid fluid in a straight, flexible tube, as depicted in Figure 1. The model is one-dimensional in an axisymmetric (r, ϕ, z) coordinate system. The non-overlapping domains of the fluid and the structure are indicated as Ω_f and Ω_s, respectively. The common boundary of these domains is the fluid-structure interface Γ, defined as

$$\Gamma = \partial\Omega_f \cap \partial\Omega_s, \tag{1}$$

with $\partial\Omega$ indicating the boundary of domain Ω.

The governing equations for the flow are the conservation of mass and momentum, formulated as

$$\frac{\partial a}{\partial t} + \frac{\partial au}{\partial z} = 0 \tag{2a}$$

$$\frac{\partial au}{\partial t} + \frac{\partial au^2}{\partial z} + \frac{1}{\rho_f}\left(\frac{\partial ap}{\partial z} - p\frac{\partial a}{\partial z}\right) = 0, \tag{2b}$$

with a the cross-sectional area of the tube, t the time, u the axial velocity and p the pressure.

For the structure, a so-called generalised string model is applied. This model is derived from the linear elasticity theory for a cylindrical tube with small thickness under the assumption of membrane deformations [1,42]. It disregards the axial and circumferential displacement of the tube wall. The governing equation for the structure is given by

$$\rho_s h \frac{\partial^2 r}{\partial t^2} - \kappa G h \frac{\partial^2 r}{\partial z^2} + \frac{Eh}{1-\nu^2} \frac{r - r_o}{r_o^2} - \gamma \frac{\partial^3 r}{\partial^2 z \partial t} = p, \tag{3}$$

with r the inner radius ($a = \pi r^2$), ρ_s the structural density and h the thickness of the tube wall. The other coefficients are the shear modulus G, the Young modulus E and the Poisson coefficient ν. The viscoelastic term is further omitted ($\gamma = 0$), which is a common simplification [42]. The Timoshenko shear correction factor κ is calculated from the Poisson coefficient [43], although it is now understood that this correction factor alone cannot always account precisely for dynamic analyses of beams with arbitrary and inhomogeneous cross sections [44].

$$\kappa = \frac{2(1+\nu)}{4+3\nu} \tag{4}$$

2.2 Discrete equations

The tube with length ℓ is discretised in space using m_e segments with length Δz. The pressure and velocity are stored in the cell centres. All terms in the flow equations are discretised using a central scheme, except for first-order upwind discretisation of the convective term in the momentum equation. The time discretisation is first-order backward Euler with time step size Δt. The discrete flow equations for each segment $m \in \{1, \ldots, m_e\}$ are given by

$$\frac{\Delta z}{\Delta t}\left(a_m - a_m^{n-1}\right) + u_{m+1/2} a_{m+1/2} - u_{m-1/2} a_{m-1/2} - \frac{\alpha}{\rho_f}\left(p_{m+1} - 2p_m + p_{m-1}\right) = 0 \tag{5a}$$

$$\frac{\Delta z}{\Delta t}\left(u_m a_m - u_m^{n-1} a_m^{n-1}\right) + u_m u_{m+1/2} a_{m+1/2} - u_{m-1} u_{m-1/2} a_{m-1/2} + \frac{1}{2\rho_f}\left[a_{m+1/2}\left(p_{m+1} - p_m\right) + a_{m-1/2}\left(p_m - p_{m-1}\right)\right] = 0 \tag{5b}$$

for $u_m > 0$. The superscript $n-1$ indicates the previous time level ($t = (n-1)\Delta t$), while the superscript n for the current time level ($t = n\Delta t$) is omitted. The velocities at the cell faces are calculated as

$$u_{m-1/2} = \frac{u_{m-1} + u_m}{2} \quad \text{and} \quad u_{m+1} = \frac{u_m + u_{m+1}}{2} \tag{6}$$

and analogously for the cross-sectional area at the cell faces.

The damping term in Equation 5a prevents spurious pressure oscillations due to the central discretisation of the pressure in the momentum equation combined with collocated pressure and velocity. Its coefficient is given by

$$\alpha = \frac{a_o}{v_o + \Delta z/\Delta t}. \tag{7}$$

This stabilisation term does not affect the accuracy of the scheme because the other terms are also first-order accurate. In [45], the suppression of pressure wiggles by this term is investigated with Fourier analysis and its implementation with higher-order accuracy on non-Cartesian grids using a finite volume discretisation is described.

To obtain structural equations with only first derivatives in time, the radial velocity of the tube wall is introduced ($v = \partial r/\partial t$). Although relatively uncommon, backward Euler time discretisation is applied for the structure as well, to avoid difficulties due to different time discretisation of the flow equations and the structural equations [46]. Moreover, the spatial discretisation is performed using a finite difference scheme instead of the typical finite elements [47]. The discrete structural equations are

$$v_m = \frac{r_m - r_m^{n-1}}{\Delta t} \tag{8a}$$

$$\rho_s h \frac{v_m - v_m^{n-1}}{\Delta t} - \kappa G h \frac{r_{m+1} - 2r_m + r_{m-1}}{\Delta z^2} + \frac{E_m h}{1-\nu^2} \frac{r_m - r_o}{r_o^2} = p_m. \tag{8b}$$

The elasticity modulus E_m of each segment is determined by the corresponding parameter s_m which varies from -1 to 1.

$$E_m = E_o \left(1 + \frac{1}{2} s_m\right) \tag{9}$$

This definition of the elasticity modulus ensures that it can vary over a realistic range. The goal during the optimisation is to identify the value of the parameters s_m ($m \in \{1, \ldots, m_e\}$) which appear in this equation.

The discretisation schemes in both time and space are only first-order accurate. However, the presented algorithms are not limited to first-order schemes. Nevertheless, their behaviour and properties could change dramatically when going to higher-order schemes. The successful implementation with higher-order accurate schemes has yet to be performed and this would be a considerable task which might not be straightforward.

2.3 Linearised equations

The above equations are subsequently linearised with respect to the reference values r_o, p_o, u_o and v_o. From this point on, r, p, u and v denote perturbations with respect to these reference values. Moreover, p_o, u_o and v_o are set to zero to simplify the resulting equations. It has been demonstrated in previous research [48] that this particular choice of the reference values results in a model with the same numerical behaviour as the model with all nonlinear

terms. Also physical properties such as wave propagation are preserved by this linearisation and reference.

The linearised flow equations are

$$\frac{\Delta z}{\Delta t}\frac{2}{r_o}\left(r_m - r_m^{n-1}\right) + \frac{1}{2}\left(u_{m+1} - u_{m-1}\right) - \frac{\Delta t}{\Delta z}\frac{1}{\rho_f}\left(p_{m+1} - 2p_m + p_{m-1}\right) = 0 \tag{10a}$$

$$\frac{\Delta z}{\Delta t}\left(u_m - u_m^{n-1}\right) + \frac{1}{2\rho_f}\left(p_{m+1} - p_{m-1}\right) = 0. \tag{10b}$$

For the structure, the linearised equations are

$$v_m = \frac{r_m - r_m^{n-1}}{\Delta t} \tag{11a}$$

$$\rho_s h\frac{v_m - v_m^{n-1}}{\Delta t} - \kappa G h\frac{r_{m+1} - 2r_m + r_{m-1}}{\Delta z^2} + \frac{E_m h}{1-\nu^2}\frac{r_m}{r_o^2} = p_m. \tag{11b}$$

2.4 Boundary conditions

As the segments of the tube are indicated with subscripts 1 to m_e, the inlet (left-hand side) is indicated with a subscript 0 and the outlet (right-hand side) with a subscript $m_e + 1$.

The blood flow rate at a point can be measured as a function of time using non-invasive techniques. So, the flow rate at the inlet is prescribed as a function of the time t with a period corresponding to one heart beat t_b.

$$u_0(t) = 0.23 + 0.21\sin\left(2\pi\frac{t}{t_b}\right) + 0.11\cos\left(4\pi\left(\frac{t}{t_b} - 0.2\right)\right) + 0.07\cos\left(6\pi\left(\frac{t}{t_b} - 0.2\right)\right) \tag{12}$$

This results in a mean flow rate of approximately $6.5{\cdot}10^{-6}\mathrm{m}^3/\mathrm{s}$ and an evolution of the inlet velocity as a function of time closely resembling the figures in [49]. The pressure at the inlet is calculated using extrapolation

$$p_0 = 2p_1 - p_2. \tag{13}$$

At the outlet of the tube, the velocity is obtained from an extrapolation

$$u_{m_e+1} = 2u_{m_e} - u_{m_e-1}, \tag{14}$$

and a Windkessel model relates this velocity with the outlet pressure [49]. This Windkessel model (see Figure 1) represents the remainder of the circulation, downstream from the artery

$$r_d q_{m_e+1} - r_d c \frac{\mathrm{d}}{\mathrm{d}t}\left(p_{m_e+1} - r_p q_{m_e+1}\right) = p_{m_e+1} - r_p q_{m_e+1}, \tag{15}$$

with $q_{m_e+1} = \pi r_o^2 u_{m_e+1}$. The subscripts p and d denote values close to (proximal) and further away (distal) from the artery, respectively. The capacitor c represents the compliance of the arterial system, while the resistors r_p and r_d model the viscous resistance. The value of c is modified by the parameter s_{m_e+1} which also varies from -1 to 1.

$$c = c_o \left(1 + \frac{1}{2} s_{m_e+1}\right)^{-1} \tag{16}$$

This definition of c is constructed analogously to Equation 9. The parameter s_{m_e+1} will be identified, together with the parameters s_m ($m \in \{1, \ldots, m_e\}$) in Equation 9.

For the structure, a zero-curvature boundary condition

$$\frac{\partial r}{\partial z} = 0 \tag{17}$$

is applied at both the inlet and the outlet.

2.5 Matrix notation

Equations 10 and Equations 11 for time step n can be written in the following block-matrix format

$$\left[\begin{array}{c|cc} \boldsymbol{M}_f & \boldsymbol{0} & \boldsymbol{0} \\ & \boldsymbol{C}_f & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{C}_s & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{M}_s \end{array}\right] \left[\begin{array}{c} \boldsymbol{x}_{f\Omega}^n \\ \boldsymbol{x}_{f\Gamma}^n \\ \hline \boldsymbol{x}_{s\Gamma}^n \\ \boldsymbol{x}_{s\Omega}^n \end{array}\right] = \left[\begin{array}{c} \boldsymbol{b}_f^n \\ \hline \boldsymbol{b}_s^n \end{array}\right] + \left[\begin{array}{c|cc} \boldsymbol{N}_f & \boldsymbol{0} & \boldsymbol{0} \\ & \boldsymbol{D}_f & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{D}_s & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{N}_s \end{array}\right] \left[\begin{array}{c} \boldsymbol{x}_{f\Omega}^{n-1} \\ \boldsymbol{x}_{f\Gamma}^{n-1} \\ \hline \boldsymbol{x}_{s\Gamma}^{n-1} \\ \boldsymbol{x}_{s\Omega}^{n-1} \end{array}\right]. \tag{18}$$

In this equation, $\boldsymbol{M}_f$ and $\boldsymbol{M}_s$ are the system matrices of the flow solver and the structural solver, respectively. The interaction between the fluid and the structure is captured by the off-diagonal blocks $\boldsymbol{C}_f$ and $\boldsymbol{C}_s$. The matrix $\boldsymbol{C}_f$ describes how the residual of the flow equations changes due to a displacement of the fluid-structure interface, whereas the matrix $\boldsymbol{C}_s$ converts the pressure on the interface into a contribution (nodal forces) to the residual of the structural equations. The state vector $\boldsymbol{x}$ is divided into a fluid part and a structure part, indicated with the subscripts f and s, respectively. Both $\boldsymbol{x}_f$ and $\boldsymbol{x}_s$ are subdivided once more using the subscripts Γ and Ω. The subscript Γ refers to variables on the fluid-structure interface, while the subscript Ω refers to variables that only appear in either the flow equations or the structural equations. The right-hand side consists of time-dependent vectors $\boldsymbol{b}_f$ and $\boldsymbol{b}_s$ which do not depend on the state $\boldsymbol{x}$, and a contribution which depends on the state in the previous time step $\boldsymbol{x}^{n-1}$.

The dimensions of $\boldsymbol{x}_{f\Omega}$, $\boldsymbol{x}_{f\Gamma}$, $\boldsymbol{x}_{s\Gamma}$ and $\boldsymbol{x}_{s\Omega}$ are respectively given by $m_{f\Omega} \times 1$, $m_{f\Gamma} \times 1$, $m_{s\Gamma} \times 1$ and $m_{s\Omega} \times 1$. The dimensions of the matrices in Equation 18

follow from the dimension of these vectors. For this specific case, the state vectors are given by

$$\boldsymbol{x}_{f\Omega} = \begin{bmatrix} u_0 \; p_0 \; u_{m_e+1} \; p_{m_e+1} \; u_1 \; u_2 \; \cdots \; u_{m_e} \end{bmatrix}^{\mathrm{T}} \tag{19a}$$

$$\boldsymbol{x}_{f\Gamma} = \begin{bmatrix} p_1 \; p_2 \; \cdots \; p_{m_e} \end{bmatrix}^{\mathrm{T}} \tag{19b}$$

$$\boldsymbol{x}_{s\Gamma} = \begin{bmatrix} r_1 \; r_2 \; \cdots \; r_{m_e} \end{bmatrix}^{\mathrm{T}} \tag{19c}$$

$$\boldsymbol{x}_{s\Omega} = \begin{bmatrix} v_1 \; v_2 \; \cdots \; v_{m_e} \end{bmatrix}^{\mathrm{T}}, \tag{19d}$$

with the superscript T indicating a transpose. The inlet and outlet model for the flow as described in Section 2.4 are included in $m_{f\Omega}$. The dimensions mentioned above can thus be written as a function of the number of tube segments m_e.

$$m_{f\Omega} = 6 + m_e \tag{20a}$$

$$m_{f\Gamma} = m_{s\Gamma} = m_{s\Omega} = m_e \tag{20b}$$

Equation 18 can be abbreviated as

$$\boldsymbol{A}\boldsymbol{x}^n = \boldsymbol{b}^n + \boldsymbol{B}\boldsymbol{x}^{n-1}, \tag{21}$$

with

$$\boldsymbol{A} = \left[\begin{array}{cc|c} & & \boldsymbol{0} \;\; \boldsymbol{0} \\ & \boldsymbol{M}_f & \boldsymbol{C}_f \;\; \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{C}_s & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{M}_s \end{array}\right] \quad \text{and} \quad \boldsymbol{B} = \left[\begin{array}{cc|c} & & \boldsymbol{0} \;\; \boldsymbol{0} \\ & \boldsymbol{N}_f & \boldsymbol{D}_f \;\; \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{D}_s & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{N}_s \end{array}\right]. \tag{22}$$

The initial state $\boldsymbol{x}^0$ is set to zero. For the specific case that is analysed in this work, the block $\boldsymbol{D}_s$ in this general equation is filled with zeros. Due to the linearisation, the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are independent of $\boldsymbol{x}$ and therefore they remain constant during the simulation. The residual of the governing equations for time step n is then given by

$$\boldsymbol{r}^n(\boldsymbol{x}^n, \boldsymbol{x}^{n-1}) = \boldsymbol{0}, \tag{23}$$

with

$$\boldsymbol{r}^n(\boldsymbol{x}^n, \boldsymbol{x}^{n-1}) = \boldsymbol{A}\boldsymbol{x}^n - \boldsymbol{b}^n - \boldsymbol{B}\boldsymbol{x}^{n-1}. \tag{24}$$

2.6 Coupling iterations

Equation 18 is solved in a partitioned way, which signifies that the flow equations and the structural equations are solved separately. Consequently, coupling iterations need to be performed between the flow equations and the structural equations to obtain the solution of the coupled problem. At convergence of the coupling iterations, the solution is identical to what a monolithic solver would calculate (up to an error proportional to the convergence tolerance of the coupling iterations). In every coupling iteration in time step n, the flow equations and the structural equations are solved with given values of $\boldsymbol{x}_{s\Gamma}$ and $\boldsymbol{x}_{f\Gamma}$, respectively. In the following sections, the superscript k indicates coupling iteration k in time step n.

At the beginning of coupling iteration k, the coupling algorithm calculates $\boldsymbol{x}_{s\Gamma}^k$, as will be explained below. The flow equations are then given by

$$\begin{bmatrix} \boldsymbol{M}_f \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{f\Omega}^k \\ \boldsymbol{x}_{f\Gamma}^k \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_f \end{bmatrix} + \begin{bmatrix} \boldsymbol{N}_f \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{f\Omega}^{n-1} \\ \boldsymbol{x}_{f\Gamma}^{n-1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{D}_f \boldsymbol{x}_{s\Gamma}^{n-1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{C}_f \boldsymbol{x}_{s\Gamma}^{k} \end{bmatrix}. \quad (25)$$

Once $\boldsymbol{x}_{f\Omega}^k$ and $\boldsymbol{x}_{f\Gamma}^k$ have been calculated, $\boldsymbol{x}_{f\Gamma}^k$ is given to the structural solver. This calculation is further referred to as $\boldsymbol{x}_{f\Gamma}^k = \boldsymbol{\mathcal{F}}(\boldsymbol{x}_{s\Gamma}^k)$.

The structural solver subsequently solves the structural equations

$$\begin{bmatrix} \boldsymbol{M}_s \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{x}}_{s\Gamma}^k \\ \boldsymbol{x}_{s\Omega}^k \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_s \end{bmatrix} + \begin{bmatrix} \boldsymbol{N}_s \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{s\Gamma}^{n-1} \\ \boldsymbol{x}_{s\Omega}^{n-1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{D}_s \boldsymbol{x}_{f\Gamma}^{n-1} \\ \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}_s \boldsymbol{x}_{f\Gamma}^{k} \\ \boldsymbol{0} \end{bmatrix} \quad (26)$$

for $\tilde{\boldsymbol{x}}_{s\Gamma}^k$ and $\boldsymbol{x}_{s\Omega}^k$. The tilde is used to distinguish between the motion of the fluid-structure interface calculated by the coupling algorithm at the beginning of the coupling iteration and that calculated by the structural solver at the end. This calculation is further referred to as $\tilde{\boldsymbol{x}}_{s\Gamma}^k = \boldsymbol{\mathcal{S}}(\boldsymbol{x}_{f\Gamma}^k)$.

The flow equations (Equation 25) are solved for given values of the motion of the fluid-structure interface, while the structural equations (Equation 26) are solved for given values of the pressure on this interface. Normally, this is called a Dirichlet-Neumann decomposition, but this name is not applicable here due to the formulation of the structural equations.

The most trivial coupling algorithm is Gauss-Seidel iteration, which means that the motion of the fluid-structure interface calculated by the structural solver at the end of the previous coupling iteration is applied by the flow solver at the beginning of the next coupling iteration.

$$\boldsymbol{x}_{s\Gamma}^k = \tilde{\boldsymbol{x}}_{s\Gamma}^{k-1} \quad (27)$$

However, it is well-understood that this coupling algorithm is unstable for fluid-structure interaction problems with an incompressible fluid and comparable density of the fluid and the structure [24,48,25].

Therefore, the interface quasi-Newton coupling algorithm with an approximation for the inverse of the Jacobian from a least-squares model (IQN-ILS) is applied [22]. This coupling algorithm treats both the flow solver and the structural solver as black boxes. Using the interface displacement applied in the flow solver ($\boldsymbol{x}_{s\Gamma}^k$) and calculated by the structural solver ($\tilde{\boldsymbol{x}}_{s\Gamma}^k$) in all coupling iterations, the coupling algorithm constructs an approximation (indicated with a hat) for the inverse of the Jacobian of the interface residual

$$\boldsymbol{r}_{s\Gamma}^k = \tilde{\boldsymbol{x}}_{s\Gamma}^k - \boldsymbol{x}_{s\Gamma}^k \tag{28}$$

with respect to the interface displacement $\boldsymbol{x}_{s\Gamma}^k$. This results in the following update at the beginning of each coupling iteration

$$\boldsymbol{x}_{s\Gamma}^k = \boldsymbol{x}_{s\Gamma}^{k-1} + \Delta\boldsymbol{x}_{s\Gamma}^k \tag{29a}$$

$$\Delta\boldsymbol{x}_{s\Gamma}^k = \widehat{\left(\frac{\partial \boldsymbol{r}_{s\Gamma}}{\partial \boldsymbol{x}_{s\Gamma}}\right)^{-1}} \Delta\boldsymbol{r}_{s\Gamma}^k, \tag{29b}$$

with $\Delta\boldsymbol{x}_{s\Gamma}^k = \boldsymbol{x}_{s\Gamma}^k - \boldsymbol{x}_{s\Gamma}^{k-1}$ and $\Delta\boldsymbol{r}_{s\Gamma}^k = \boldsymbol{r}_{s\Gamma}^k - \boldsymbol{r}_{s\Gamma}^{k-1} = -\boldsymbol{r}_{s\Gamma}^{k-1}$ because the goal is to find $\boldsymbol{x}_{s\Gamma}^k$ so that $\boldsymbol{r}_{s\Gamma}^k = \boldsymbol{0}$.

An approximation for the inverse of the Jacobian will lead to convergence of the coupling iterations if it contains the wave numbers in the interface displacement that are unstable during Gauss-Seidel iterations. For the flow in a one-dimensional flexible tube, it has been shown analytically that only a fraction of these wave numbers are unstable [25]. Other problems have shown similar behaviour. Therefore, the IQN-ILS algorithm constructs a low-rank approximation for the inverse of this Jacobian matrix using least-squares. As only a fraction of the wave numbers are unstable, a full-rank matrix is not required. For the wave numbers that are not included in this least-squares approximation, IQN-ILS iterations correspond with Gauss-Seidel iterations. In every coupling iteration, a rank-one update is applied to the approximation for the inverse of the Jacobian so that this approximation improves. It has been demonstrated that this least-squares approach is faster for interaction problems than other rank-one update methods such as Broyden's method [50].

Because only the product of the approximation for the inverse of the Jacobian with a vector is required, this matrix does not have to be constructed explicitly. In a matrix-free implementation of IQN-ILS, the computational cost and the memory requirements of the coupling algorithm scale linearly with the number of degrees of freedom in the interface displacement ($m_{s\Gamma}$). In the following paragraphs, the different steps are explained in detail.

At the beginning of the first coupling iteration ($k = 1$), the result of the previous time steps is extrapolated. The order of the extrapolation depends on how many time steps have been performed.

$$\boldsymbol{x}_{s\Gamma}^{k} = \boldsymbol{\mathcal{E}}^{n}(\boldsymbol{x}_{s\Gamma}^{n-1}, \boldsymbol{x}_{s\Gamma}^{n-2}, \boldsymbol{x}_{s\Gamma}^{n-3}) = \begin{cases} n=1: & \boldsymbol{x}_{s\Gamma}^{n-1} \\ n=2: & 2\boldsymbol{x}_{s\Gamma}^{n-1} - \boldsymbol{x}_{s\Gamma}^{n-2} \\ n>2: & \frac{5}{2}\boldsymbol{x}_{s\Gamma}^{n-1} - 2\boldsymbol{x}_{s\Gamma}^{n-2} + \frac{1}{2}\boldsymbol{x}_{s\Gamma}^{n-3} \end{cases} \tag{30}$$

At the beginning of the second coupling iteration ($k = 2$), the change in $\boldsymbol{x}$ is relaxed with a fixed factor ω. Without reuse of information from previous time steps, no quasi-Newton step can be done in the second coupling iteration because the IQN-ILS coupling algorithm requires the information from at least two coupling iterations.

$$\boldsymbol{x}_{s\Gamma}^{k} = \boldsymbol{x}_{s\Gamma}^{k-1} + \omega \boldsymbol{r}_{s\Gamma}^{k-1} = (1-\omega)\boldsymbol{x}_{s\Gamma}^{k-1} + \omega \tilde{\boldsymbol{x}}_{s\Gamma}^{k-1} \tag{31}$$

This relaxation step is skipped if columns from previous time steps are reused in the least-squares model. The only requirement for the relaxation parameter ω is that it avoids excessive divergence in the second coupling iteration, which could cause errors in the solvers, for example due to a corrupted mesh. Previous research has shown that the IQN-ILS method is robust with respect to this parameter and that it does not influence the long-term convergence [51]. A typical value is $\omega = 10^{-2}$.

At the beginning of all subsequent coupling iterations ($k > 2$), the IQN-ILS algorithm calculates the value of $\boldsymbol{x}_{s\Gamma}^{k}$ based on the information from the $k-1$ previous coupling iterations. The differences between consecutive coupling iterations ($i \in \{1, \ldots, k-2\}$)

$$\Delta \boldsymbol{r}_{s\Gamma}^{i} = \boldsymbol{r}_{s\Gamma}^{i+1} - \boldsymbol{r}_{s\Gamma}^{i} \tag{32a}$$

$$\Delta \tilde{\boldsymbol{x}}_{s\Gamma}^{i} = \tilde{\boldsymbol{x}}_{s\Gamma}^{i+1} - \tilde{\boldsymbol{x}}_{s\Gamma}^{i}. \tag{32b}$$

are calculated and stored as columns of the matrices $\boldsymbol{V}^k$ and $\boldsymbol{W}^k$, giving

$$\boldsymbol{V}^{k} = \left[\Delta \boldsymbol{r}_{s\Gamma}^{k-2} \ \ldots \ \Delta \boldsymbol{r}_{s\Gamma}^{2} \ \Delta \boldsymbol{r}_{s\Gamma}^{1}\right] \tag{33a}$$

$$\boldsymbol{W}^{k} = \left[\Delta \tilde{\boldsymbol{x}}_{s\Gamma}^{k-2} \ \ldots \ \Delta \tilde{\boldsymbol{x}}_{s\Gamma}^{2} \ \Delta \tilde{\boldsymbol{x}}_{s\Gamma}^{1}\right]. \tag{33b}$$

Both matrices have as dimension $m_{s\Gamma} \times k-2$, unless the columns from a number of time steps are reused to accelerate the convergence.

The change of the residual $\Delta \boldsymbol{r}_{s\Gamma}^{k} = -\boldsymbol{r}_{s\Gamma}^{k-1}$ that is required to go from the last residual $\boldsymbol{r}_{s\Gamma}^{k-1}$ to $\boldsymbol{0}$ is decomposed as a linear combination of the known changes of the residual

$$\Delta \boldsymbol{r}_{s\Gamma}^{k} = \boldsymbol{V}^{k} \boldsymbol{c}^{k}, \tag{34}$$

with $\boldsymbol{c}^{k} \in \mathbb{R}^{k-2 \times 1}$ the vector of decomposition coefficients. As there are usually more than $k-2$ degrees of freedom in the interface displacement ($m_{s\Gamma} > k-2$), this system is overdetermined. It can be solved for $\boldsymbol{c}^{k}$ using least-squares, for example using the QR-decomposition of $\boldsymbol{V}^{k}$,

$$\boldsymbol{V}^{k} = \boldsymbol{Q}^{k} \boldsymbol{R}^{k}, \tag{35}$$

with $\boldsymbol{Q}^k \in \mathbb{R}^{m_{s\Gamma} \times k-2}$ an orthogonal matrix and $\boldsymbol{R}^k \in \mathbb{R}^{k-2 \times k-2}$ an upper triangular matrix. The decomposition coefficients are then obtained using back-substitution in the upper triangular system

$$\boldsymbol{R}^k \boldsymbol{c}^k = \left(\boldsymbol{Q}^k\right)^{\mathrm{T}} \Delta \boldsymbol{r}_{s\Gamma}^k. \tag{36}$$

Because column i of $\boldsymbol{V}^k$ corresponds with column i of $\boldsymbol{W}^k$, it is assumed that a linear combination of the columns of $\boldsymbol{V}^k$ corresponds with the same linear combination of the columns of $\boldsymbol{W}^k$. As a result, the $\Delta \tilde{\boldsymbol{x}}_{s\Gamma}^k$ that corresponds with $\Delta \boldsymbol{r}_{s\Gamma}^k$ is given by

$$\Delta \tilde{\boldsymbol{x}}_{s\Gamma}^k = \boldsymbol{W}^k \boldsymbol{c}^k. \tag{37}$$

The calculation of $\Delta \tilde{\boldsymbol{x}}_{s\Gamma}^k$ for a given $\Delta \boldsymbol{r}_{s\Gamma}^k$ (as carried out in Equation 36 and Equation 37) is further abbreviated as $\Delta \tilde{\boldsymbol{x}}_{s\Gamma}^k = \boldsymbol{\mathcal{M}}^k(\Delta \boldsymbol{r}_{s\Gamma}^k)$. Using the definition of the interface residual (Equation 28), the sought-after $\Delta \boldsymbol{x}_{s\Gamma}^k$ is finally given by

$$\Delta \boldsymbol{x}_{s\Gamma}^k = \Delta \tilde{\boldsymbol{x}}_{s\Gamma}^k - \Delta \boldsymbol{r}_{s\Gamma}^k = \boldsymbol{\mathcal{M}}^k(-\boldsymbol{r}_{s\Gamma}^{k-1}) + \boldsymbol{r}_{s\Gamma}^{k-1} \tag{38}$$

Coupling iterations are performed until $||\boldsymbol{r}_{s\Gamma}^k||_2 < \epsilon_c ||\boldsymbol{r}_{s\Gamma}^1||_2$, in which ϵ_c denotes the relative convergence criterion of the coupling iterations. The complete procedure for the forward simulation is described in Algorithm 1.

Algorithm 1 The IQN-ILS coupling algorithm for the forward simulation.

for $n = 1, \ldots, n_e$ **do**
  for $k = 1, \ldots, k_e$ **do**
    if $k = 1$ **then**
      $\boldsymbol{x}_{s\Gamma}^k = \boldsymbol{\mathcal{E}}^n(\boldsymbol{x}_{s\Gamma}^{n-1}, \boldsymbol{x}_{s\Gamma}^{n-2}, \boldsymbol{x}_{s\Gamma}^{n-3})$ Equation 30
    else if $k = 2$ and no reuse **then**
      $\boldsymbol{x}_{s\Gamma}^k = \boldsymbol{x}_{s\Gamma}^{k-1} + \omega \boldsymbol{r}_{s\Gamma}^{k-1}$ Equation 31
    else
      $\Delta \boldsymbol{x}_{s\Gamma}^k = \boldsymbol{\mathcal{M}}^k(-\boldsymbol{r}_{s\Gamma}^{k-1}) + \boldsymbol{r}_{s\Gamma}^{k-1}$ Equation 38
      $\boldsymbol{x}_{s\Gamma}^k = \boldsymbol{x}_{s\Gamma}^{k-1} + \Delta \boldsymbol{x}_{s\Gamma}^k$
    end if
    $\boldsymbol{x}_{f\Gamma}^k = \boldsymbol{\mathcal{F}}(\boldsymbol{x}_{s\Gamma}^k)$ Equation 25
    $\tilde{\boldsymbol{x}}_{s\Gamma}^k = \boldsymbol{\mathcal{S}}(\boldsymbol{x}_{f\Gamma}^k)$ Equation 26
    $\boldsymbol{r}_{s\Gamma}^k = \tilde{\boldsymbol{x}}_{s\Gamma}^k - \boldsymbol{x}_{s\Gamma}^k$ Equation 28
    if $k > 2$ and $||\boldsymbol{r}_{s\Gamma}^k||_2 < \epsilon_c ||\boldsymbol{r}_{s\Gamma}^1||_2$ **then**
      $k = k_e + 1$
    end if
  end for
end for

3 Sensitivity analysis

3.1 Cost function

The cost function j is defined using a normalised difference between a measurement and a simulation of the model described above. In this work, this measurement, which would normally be obtained from a non-invasive medical imaging technique, is mimicked by a simulation with the same model. It is then assumed that the parameter values in this "measurement simulation" have been forgotten and their values are calculated using the parameter identification. This makes it possible to simplify the implementation and to verify the procedure by comparing the parameter values obtained from the parameter identification with those used during the "measurement simulation". This also means that the cost function will be zero when the correct parameters have been found.

The dimensionless least-squares cost function does not contain regularisation terms. It is given by

$$j(\boldsymbol{s},\boldsymbol{x}) = \frac{\left(\bar{\boldsymbol{x}}_{s\Gamma} - \bar{\boldsymbol{x}}_{s\Gamma}^{ref}\right)^{\mathrm{T}} \left(\bar{\boldsymbol{x}}_{s\Gamma} - \bar{\boldsymbol{x}}_{s\Gamma}^{ref}\right)}{m_e n_e \left(\max \bar{\boldsymbol{x}}_{s\Gamma}^{ref} - \min \bar{\boldsymbol{x}}_{s\Gamma}^{ref}\right)^2}, \tag{39}$$

with the superscript ref referring to the measurement (or reference). The vector $\boldsymbol{s}$ contains the $m_e + 1$ parameters which are defined in Equation 9 and Equation 16. The vector $\boldsymbol{x}$ is the combination of the state vectors of all time steps

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \\ \vdots \\ \boldsymbol{x}^{n_e} \end{bmatrix}. \tag{40}$$

The vector $\bar{\boldsymbol{x}}_{s\Gamma}$ is identical to $\boldsymbol{x}$, but all entries which do not correspond to $\boldsymbol{x}_{s\Gamma}^n$ have been set to zero, giving

$$\bar{\boldsymbol{x}}_{s\Gamma} = \begin{bmatrix} \bar{\boldsymbol{x}}_{s\Gamma}^1 \\ \bar{\boldsymbol{x}}_{s\Gamma}^2 \\ \vdots \\ \bar{\boldsymbol{x}}_{s\Gamma}^{n_e} \end{bmatrix}, \quad \text{with} \quad \bar{\boldsymbol{x}}_{s\Gamma}^n = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \hline \boldsymbol{x}_{s\Gamma}^n \\ \mathbf{0} \end{bmatrix} \cdot \begin{matrix} \} \, m_{f\Omega} \times 1 \\ \} \, m_{f\Gamma} \times 1 \\ \} \, m_{s\Gamma} \times 1 \\ \} \, m_{s\Omega} \times 1 \end{matrix} \tag{41}$$

So, the cost function is a sum over all time steps and all tube segments of the squared difference between the radius in the simulation and in the measurement.

3.2 Minimisation problem

With the definition of the cost function in Equation 39, the parameter identification can be reformulated as a minimisation problem

$$\min_{\boldsymbol{s},\boldsymbol{x}} j(\boldsymbol{s},\boldsymbol{x}) \tag{42}$$

subject to the governing equations as constraints

$$\boldsymbol{r}(\boldsymbol{s},\boldsymbol{x}) = \boldsymbol{0}. \tag{43}$$

The parameter vector $\boldsymbol{s}$ and the state vector $\boldsymbol{x}$ are defined in Equations 9, 16 and 40. The cost function of the parameter identification is the only objective function of the minimisation. Here, $\boldsymbol{r}$ is the combination of the residual of the governing equations in all time steps

$$\boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}^1 \\ \boldsymbol{r}^2 \\ \vdots \\ \boldsymbol{r}^{n_e} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}^1(\boldsymbol{x}^1, \boldsymbol{x}^0) \\ \boldsymbol{r}^2(\boldsymbol{x}^2, \boldsymbol{x}^1) \\ \vdots \\ \boldsymbol{r}^{n_e}(\boldsymbol{x}^{n_e}, \boldsymbol{x}^{n_e-1}) \end{bmatrix}. \tag{44}$$

3.3 Adjoint equations

As the state vector depends on the parameters, the gradient of the cost function $j(\boldsymbol{s},\boldsymbol{x}) = j(\boldsymbol{s},\boldsymbol{x}(\boldsymbol{s}))$ requires application of the chain rule. The total derivatives of j with respect to the parameters are

$$\frac{\mathrm{d}j}{\mathrm{d}\boldsymbol{s}} = \frac{\partial j}{\partial \boldsymbol{s}} + \frac{\partial j}{\partial \boldsymbol{x}} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{s}}. \tag{45}$$

The partial derivatives in this equation can be calculated quickly and easily from Equation 39, as the state $\boldsymbol{x}$ remains constant for $\partial j/\partial \boldsymbol{s}$ and the parameters $\boldsymbol{s}$ remain constant for $\partial j/\partial \boldsymbol{x}$. By contrast, the total derivative $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{s}$ requires the solution of the unsteady fluid-structure interaction problem and it is thus time-consuming to calculate.

As the governing equations of the fluid-structure interaction problem always need to be satisfied, $\boldsymbol{r}$ should always be equal to $\boldsymbol{0}$. Consequently, also the total derivative $\mathrm{d}\boldsymbol{r}/\mathrm{d}\boldsymbol{s}$ should be equal to $\boldsymbol{0}$. By applying the chain rule, this total derivative is given by

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\boldsymbol{s}} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{s}} + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{s}} = \boldsymbol{0}. \tag{46}$$

Rewriting the previous equation as

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{s}} = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{s}} \tag{47}$$

results in a system that can be solved for the total derivative $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{s}$. The result is subsequently substituted in Equation 45, yielding

$$\frac{\mathrm{d}j}{\mathrm{d}\boldsymbol{s}} = \frac{\partial j}{\partial \boldsymbol{s}} - \underbrace{\frac{\partial j}{\partial \boldsymbol{x}} \overbrace{\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}\right)^{-1}}^{-\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{s}}}_{-\boldsymbol{a}^{\mathrm{T}}} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{s}}. \tag{48}$$

In this equation, the matrix inversion is a symbolic notation for the solution of a large system, which combines all equations of all time steps of the unsteady fluid-structure interaction problem. Obviously, this matrix is neither constructed explicitly nor inverted or factorised due to excessive memory requirements. As explained above, the time steps are calculated consecutively with a partitioned solution technique.

Depending on how the factors are grouped in the last term of Equation 48, two methods can be distinguished. The direct method solves Equation 47 for $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{s}$ and substitutes the result in Equation 48. In this method, the product between the last two factors in the last term of Equation 48

$$\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}\right)^{-1} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{s}} \tag{49}$$

is thus calculated first. Afterwards, $\partial j/\partial \boldsymbol{x}$ is multiplied with the result. Since the right-hand side of Equation 47 consists of as many columns as there are parameters, this approach corresponds with the solution of a linear system for every parameter. If a factorisation of the matrix $\partial \boldsymbol{r}/\partial \boldsymbol{x}$ can be stored, the computational cost can be acceptable as only a back substitution needs to be performed for every column of the right-hand side. However, in the case of unsteady fluid-structure interaction, this matrix is too large to be factorised and stored. As a result, the computational cost of solving Equation 47 is similar to the cost of solving the governing equations, multiplied by the number of parameters. Hence, the direct approach is only suitable for a small number of parameters. The advantage of this method is that $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{s}$ does not have to be recalculated if several objective functions need to be minimised.

By contrast, the adjoint method first calculates the product of the first two factors in the last term of Equation 48.

$$\frac{\partial j}{\partial \boldsymbol{x}} \left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}\right)^{-1} = -\boldsymbol{a}^{\mathrm{T}} \tag{50}$$

The vector $\boldsymbol{a}$ is the so-called adjoint (or dual) state, which is the solution of the adjoint equation

$$\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}\right)^{\mathrm{T}} \boldsymbol{a} = -\left(\frac{\partial j}{\partial \boldsymbol{x}}\right)^{\mathrm{T}}. \tag{51}$$

The sought-after gradient $\mathrm{d}j/\mathrm{d}\boldsymbol{s}$ is finally calculated as

$$\frac{\mathrm{d}j}{\mathrm{d}\boldsymbol{s}} = \frac{\partial j}{\partial \boldsymbol{s}} + \boldsymbol{a}^{\mathrm{T}} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{s}}. \tag{52}$$

The computational cost of solving Equation 51 is independent of the number of parameters, so the adjoint method is suitable for a large number of parameters. However, there are as many right-hand sides as objective functions. As for the direct method, the matrix in Equation 51 is too large to be factorised and stored in the case of unsteady fluid-structure interaction. So, the adjoint equations need to be solved for every objective function. Alternatively, the adjoint method can be derived by introducing Lagrange multipliers, for example as in [52].

From the explanation above, it is clear that the choice between the direct and the adjoint method depends on the number of parameters and objective functions. The direct method requires the solution of a problem comparable to the governing equations for every parameter, while its cost is almost independent of the number of objective functions. The opposite is true for the adjoint method. If central (respectively forward) finite differences are used for the calculation of the gradient, the governing equations need to be solved twice (respectively once) for every parameter and for every objective function. In this specific case, there is only one objective function and there are many parameters, so the adjoint method is selected.

In this case, the cost function (Equation 39) does not explicitly depend on the parameters $\boldsymbol{s}$, so

$$\frac{\partial j}{\partial \boldsymbol{s}} = \boldsymbol{0}. \tag{53}$$

Conversely, the residual $\boldsymbol{r}$ (Equation 24) depends on the parameters $\boldsymbol{s}$, giving

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{s}} = \begin{bmatrix} \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{s}}\boldsymbol{x}^1 - \left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{s}}\right)^1 - \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{s}}\boldsymbol{x}^0 \\ \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{s}}\boldsymbol{x}^2 - \left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{s}}\right)^2 - \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{s}}\boldsymbol{x}^1 \\ \vdots \\ \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{s}}\boldsymbol{x}^{n_e} - \left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{s}}\right)^{n_e} - \frac{\partial \boldsymbol{B}}{\partial \boldsymbol{s}}\boldsymbol{x}^{n_e-1} \end{bmatrix}. \tag{54}$$

The only non-zero contributions to $\partial \boldsymbol{r}/\partial \boldsymbol{s}$ are due to the terms that contain the elasticity modulus E_m (Equation 9) and the compliance of the proximal arteries c (Equation 16).

$$\frac{\partial E_m}{\partial s_m} = \frac{E_o}{2} \tag{55a}$$

$$\frac{\partial c}{\partial s_{m_e+1}} = \frac{-c_o}{\left(1 + \frac{1}{2}s_{m_e+1}\right)^2}\frac{1}{2} \tag{55b}$$

In the adjoint equation (Equation 51), both $(\partial j/\partial \boldsymbol{x})^{\mathrm{T}}$ and $(\partial \boldsymbol{r}/\partial \boldsymbol{x})^{\mathrm{T}}$ are required. The former is calculated analytically, giving

$$\left(\frac{\partial j}{\partial \boldsymbol{x}}\right)^{\mathrm{T}} = \frac{2\left(\bar{\boldsymbol{x}}_{s\Gamma} - \bar{\boldsymbol{x}}_{s\Gamma}^{ref}\right)}{m_e n_e \left(\max \bar{\boldsymbol{x}}_{s\Gamma}^{ref} - \min \bar{\boldsymbol{x}}_{s\Gamma}^{ref}\right)^2}. \tag{56}$$

The calculation of $\partial \boldsymbol{r}/\partial \boldsymbol{x}$ and its transpose are more involved. This matrix is not constructed or stored, but the solution to Equation 51 is calculated using time steps. Using Equation 24, $\partial \boldsymbol{r}/\partial \boldsymbol{x}$ is given by

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{r}^1}{\partial \boldsymbol{x}^1} & \frac{\partial \boldsymbol{r}^1}{\partial \boldsymbol{x}^2} & \cdots & \frac{\partial \boldsymbol{r}^1}{\partial \boldsymbol{x}^{n_e}} \\ \frac{\partial \boldsymbol{r}^2}{\partial \boldsymbol{x}^1} & \frac{\partial \boldsymbol{r}^2}{\partial \boldsymbol{x}^2} & & \frac{\partial \boldsymbol{r}^2}{\partial \boldsymbol{x}^{n_e}} \\ \vdots & & \ddots & \\ \frac{\partial \boldsymbol{r}^{n_e}}{\partial \boldsymbol{x}^1} & \frac{\partial \boldsymbol{r}^{n_e}}{\partial \boldsymbol{x}^2} & & \frac{\partial \boldsymbol{r}^{n_e}}{\partial \boldsymbol{x}^{n_e}} \end{bmatrix} \tag{57a}$$

$$= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{B} & \boldsymbol{A} & \boldsymbol{0} & & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{B} & \boldsymbol{A} & & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & & & \ddots & & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & & -\boldsymbol{B} & \boldsymbol{A} \end{bmatrix} \tag{57b}$$

Because $\partial \boldsymbol{r}/\partial \boldsymbol{x}$ is lower bidiagonal, the forward simulation consists of forward time steps. By contrast, its transpose is upper bidiagonal, which is the reason for the backward time steps in the adjoint simulation.

$$\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}\right)^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{A}^{\mathrm{T}} & -\boldsymbol{B}^{\mathrm{T}} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^{\mathrm{T}} & -\boldsymbol{B}^{\mathrm{T}} & & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}^{\mathrm{T}} & & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & & & \ddots & & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{A}^{\mathrm{T}} & -\boldsymbol{B}^{\mathrm{T}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{0} & \boldsymbol{A}^{\mathrm{T}} \end{bmatrix} \tag{58}$$

3.4 Matrix notation

Considering Equation 58, the block-matrix format of a time step for the solution of the adjoint equation (Equation 51) is similar to Equation 18.

$$\left[\begin{array}{cc|cc} \boldsymbol{M}_f^{\mathrm{T}} & & \boldsymbol{0} & \boldsymbol{0} \\ & & \boldsymbol{C}_s^{\mathrm{T}} & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{C}_f^{\mathrm{T}} & & \\ \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{M}_s^{\mathrm{T}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{a}_{f\Omega}^n \\ \boldsymbol{a}_{f\Gamma}^n \\ \hline \boldsymbol{a}_{s\Gamma}^n \\ \boldsymbol{a}_{s\Omega}^n \end{array}\right] = - \left[\begin{array}{c} \left(\frac{\partial j}{\partial \boldsymbol{x}_f^n}\right)^{\mathrm{T}} \\ \hline \left(\frac{\partial j}{\partial \boldsymbol{x}_s^n}\right)^{\mathrm{T}} \end{array}\right] + \left[\begin{array}{cc|cc} \boldsymbol{N}_f^{\mathrm{T}} & & \boldsymbol{0} & \boldsymbol{0} \\ & & \boldsymbol{D}_s^{\mathrm{T}} & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{D}_f^{\mathrm{T}} & & \\ \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{N}_s^{\mathrm{T}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{a}_{f\Omega}^{n+1} \\ \boldsymbol{a}_{f\Gamma}^{n+1} \\ \hline \boldsymbol{a}_{s\Gamma}^{n+1} \\ \boldsymbol{a}_{s\Omega}^{n+1} \end{array}\right] \tag{59}$$

This equation is written in a general form. The structure is self-adjoint which allows for a simplification ($\boldsymbol{M}_s^{\mathrm{T}} = \boldsymbol{M}_s$), but this is not used here to keep the formulation general. In abbreviated form, the previous equation yields

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{a}^n = -\left(\frac{\partial j}{\partial \boldsymbol{x}^n}\right)^{\mathrm{T}} + \boldsymbol{B}^{\mathrm{T}}\boldsymbol{a}^{n+1}, \tag{60}$$

which is similar to Equation 21, except for the backward time steps ($n \in \{n_e, \ldots, 1\}$). The initial adjoint state $\boldsymbol{a}^{n_e+1}$ is again set to zero.

3.5 Coupling iterations

Equation 59 is solved in a partitioned way as well, by performing coupling iterations between the adjoint flow equations and the adjoint structural equations until the solution of the coupled adjoint problem has been found. The superscript k again indicates coupling iteration k in time step n.

The adjoint flow equations in coupling iteration k are given by

$$\begin{bmatrix} \boldsymbol{M}_f^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_{f\Omega}^k \\ \boldsymbol{a}_{f\Gamma}^k \end{bmatrix} = -\begin{bmatrix} \left(\frac{\partial j}{\partial \boldsymbol{x}_f^n}\right)^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{N}_f^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_{f\Omega}^{n+1} \\ \boldsymbol{a}_{f\Gamma}^{n+1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{D}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{k} \end{bmatrix}, \tag{61}$$

which are solved for $\boldsymbol{a}_{f\Omega}^k$ and $\boldsymbol{a}_{f\Gamma}^k$. This equation contains $\boldsymbol{C}_s$ and $\boldsymbol{D}_s$, which belong to the structural solver. The exchange of matrices between the flow solver and the structural solver is highly unwanted in the partitioned approach. Therefore, $\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k$ and $\boldsymbol{D}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+1}$ are given to the flow solver, instead of $\boldsymbol{a}_{s\Gamma}^k$ and $\boldsymbol{a}_{s\Gamma}^{n+1}$. While $\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k$ is provided to the flow solver in every coupling iteration, $\boldsymbol{D}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+1}$ is only exchanged once at the beginning of every time step. In the special case that $\boldsymbol{C}_s$ and $\boldsymbol{D}_s$ are identical, giving $\boldsymbol{D}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+1}$ to the flow solver is not required because the flow solver can store the information from t^{n+1}.

The structural solver subsequently solves the adjoint structural equations

$$\begin{bmatrix} \boldsymbol{M}_s^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{a}}_{s\Gamma}^k \\ \boldsymbol{a}_{s\Omega}^k \end{bmatrix} = -\begin{bmatrix} \left(\frac{\partial j}{\partial \boldsymbol{x}_s^n}\right)^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{N}_s^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_{s\Gamma}^{n+1} \\ \boldsymbol{a}_{s\Omega}^{n+1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{D}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^{n+1} \\ \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^{k} \\ \boldsymbol{0} \end{bmatrix} \tag{62}$$

for $\tilde{\boldsymbol{a}}_{s\Gamma}^k$ and $\boldsymbol{a}_{s\Omega}^k$. The tilde is used to distinguish between the adjoint state calculated by the coupling algorithm at the beginning of the coupling iteration and that calculated by the structural solver at the end. This equation contains $\boldsymbol{C}_f$ and $\boldsymbol{D}_f$, which belong to the flow solver. To avoid the exchange of matrices, $\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^k$ and $\boldsymbol{D}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^{n+1}$ are provided to the structural solver. While the former is

given in every coupling iteration, the latter is only given once at the beginning of the time step.

In conclusion, the flow solver calculates $\boldsymbol{a}_f^k$ for a given $\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k$ and gives $\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^k$ to the structural solver, whereas the structural solver calculates $\boldsymbol{a}_s^k$ for a given $\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^k$ and returns $\boldsymbol{C}_s^{\mathrm{T}}\tilde{\boldsymbol{a}}_{s\Gamma}^k$. These calculations are further referred to as $\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^k = \boldsymbol{\mathcal{F}}^{\mathrm{T}}(\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k)$ and $\boldsymbol{C}_s^{\mathrm{T}}\tilde{\boldsymbol{a}}_{s\Gamma}^k = \boldsymbol{\mathcal{S}}^{\mathrm{T}}(\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^k)$.

The adjoint flow equations (Equation 61) and structural equations (Equation 62) are coupled using the IQN-ILS algorithm, similarly to the forward equations. The partitioned adjoint simulation is described in Algorithm 2. The differences with the partitioned forward simulation are as follows. First, the vectors $\boldsymbol{D}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^{n+1}$ and $\boldsymbol{D}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+1}$ are exchanged between the flow solver and the structural solver at the beginning of every time step (lines 2-3). Then, the extrapolation ($\boldsymbol{\mathcal{E}}$) is adapted in a straightforward way to the backward time steps (line 6). Moreover, the interface residual is defined as

$$\boldsymbol{r}_{s\Gamma}^k = \boldsymbol{C}_s^{\mathrm{T}}\tilde{\boldsymbol{a}}_{s\Gamma}^k - \boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k \tag{63}$$

in the adjoint simulation. Finally, $\tilde{\boldsymbol{x}}_{s\Gamma}^i$ is replaced by $\boldsymbol{C}_s^{\mathrm{T}}\tilde{\boldsymbol{a}}_{s\Gamma}^i$ in the least-squares model ($\boldsymbol{\mathcal{M}}$ on line 10).

Algorithm 2 The IQN-ILS coupling algorithm for the adjoint simulation.

1: **for** $n = n_e, \ldots, 1$ **do**
2: send $\boldsymbol{D}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^{n+1}$ from flow solver to structural solver
3: send $\boldsymbol{D}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+1}$ from structural solver to flow solver
4: **for** $k = 1, \ldots, k_e$ **do**
5: **if** $k = 1$ **then**
6: $\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k = \boldsymbol{\mathcal{E}}^n(\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+1}, \boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+2}, \boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{n+3})$ Equation 30
7: **else if** $k = 2$ and no reuse **then**
8: $\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k = \boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{k-1} + \omega\boldsymbol{r}_{s\Gamma}^{k-1}$ Equation 31
9: **else**
10: $\Delta\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k = \boldsymbol{\mathcal{M}}^k(-\boldsymbol{r}_{s\Gamma}^{k-1}) + \boldsymbol{r}_{s\Gamma}^{k-1}$ Equation 38
11: $\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k = \boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^{k-1} + \Delta\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k$
12: **end if**
13: $\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^k = \boldsymbol{\mathcal{F}}^{\mathrm{T}}(\boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k)$ Equation 61
14: $\boldsymbol{C}_s^{\mathrm{T}}\tilde{\boldsymbol{a}}_{s\Gamma}^k = \boldsymbol{\mathcal{S}}^{\mathrm{T}}(\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{a}_{f\Gamma}^k)$ Equation 62
15: $\boldsymbol{r}_{s\Gamma}^k = \boldsymbol{C}_s^{\mathrm{T}}\tilde{\boldsymbol{a}}_{s\Gamma}^k - \boldsymbol{C}_s^{\mathrm{T}}\boldsymbol{a}_{s\Gamma}^k$ Equation 63
16: **if** $k > 2$ and $||\boldsymbol{r}_{s\Gamma}^k||_2 < \epsilon_c||\boldsymbol{r}_{s\Gamma}^1||_2$ **then**
17: $k = k_e + 1$
18: **end if**
19: **end for**
20: **end for**

4 Optimisation

For the minimisation of the cost function j (Equation 42), unconstrained quasi-Newton optimisation with line search is applied. Only a brief overview of the

optimisation strategy is provided, as it is described in detail in [53]. In theory, the parameters should be constrained so that the physical parameters remain positive. However, it has been observed that this requirement is fulfilled during unconstrained optimisation, so no constraints have been imposed.

In the quasi-Newton optimisation, the parameters $\boldsymbol{s}$ are calculated as

$$\boldsymbol{s}_\ell = \boldsymbol{s}_{\ell-1} + \alpha_{\ell-1}\boldsymbol{d}_{\ell-1}, \tag{64}$$

with the search direction $\boldsymbol{d}$ defined as

$$\boldsymbol{d}_{\ell-1} = -\boldsymbol{H}_{\ell-1}\left(\frac{\mathrm{d}j}{\mathrm{d}\boldsymbol{s}}\right)^{\mathrm{T}}_{\ell-1}. \tag{65}$$

In these equations, the subscript ℓ denotes the optimisation iteration, α the step length and $\boldsymbol{H}$ the approximation for the inverse of the cost function's Hessian. The optimisation halts when the first-order optimality criterion

$$||\left(\mathrm{d}j/\mathrm{d}\boldsymbol{s}\right)_\ell||_\infty < \epsilon_o(1 + ||\left(\mathrm{d}j/\mathrm{d}\boldsymbol{s}\right)_1||_\infty) \tag{66}$$

is satisfied, with ϵ_o the relative convergence tolerance of the optimisation iterations. Alternatively, the optimisation stops when the step size is too small.

$$||(\boldsymbol{s}_\ell - \boldsymbol{s}_{\ell-1})/(1 + |\boldsymbol{s}_\ell|)||_\infty < \epsilon_s \tag{67}$$

The goal of the one-dimensional line search is to find the step length ($\alpha > 0$) which minimises the function

$$\phi(\alpha_{\ell=1}) = j(\boldsymbol{s}_{\ell-1} + \alpha_{\ell-1}\boldsymbol{d}_{\ell-1}), \tag{68}$$

with as few evaluations as possible of the cost function and its gradient. As finding the true minimum would be too time-consuming, this is reduced to finding a step length which satisfies the strong Wolfe conditions

$$\phi(\alpha_{\ell-1}) \leq \phi(0) + c_1\alpha_{\ell-1}\phi'(0) \tag{69a}$$

$$|\phi(\alpha_{\ell-1})'| \leq c_2|\phi(0)'|, \tag{69b}$$

with $0 < c_1 < c_2 < 1$ and ϕ' the derivative of ϕ with respect to α. The first condition enforces a reduction of the cost function, the second one that the step length lies in at least a broad neighbourhood of a local minimiser or stationary point of ϕ. In the bracketing phase of the line search, an interval of acceptable step lengths is determined. This is followed by a selection phase which locates the final step length using gradual reduction of this interval's size and cubic interpolation based on ϕ and ϕ' at the two last values of α.

The approximation for the inverse of the cost function's Hessian is updated after every optimisation iteration using the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm. This update is based on the new value of the parameters and the cost function's gradient. A parameter determines for how many optimisation iterations these two vectors are stored. As the inverse of the Hessian (instead of the Hessian itself) is approximated, no matrix

factorisation is needed. Moreover, the approximation for the inverse of the Hessian is not stored explicitly, but the two-loop recursion algorithm is applied to compute its product with the cost function's gradient (Equation 65). As a result, the memory requirement of the optimisation is linear (and not quadratic) in the number of optimisation parameters.

The overview of the complete parameter identification computation is shown in Algorithm 3. This overview facilitates the understanding of the implementation, as described in the following section.

Algorithm 3 The overview of the complete parameter identification computation.

1:	**for** $\ell = 1, \ldots, \ell_e$ **do**	
2:	**if** $\ell = 1$ **then**	
3:	$\boldsymbol{s}_\ell = \boldsymbol{s}_o$	
4:	**else**	
5:	calculate search direction $\boldsymbol{d}_{\ell-1}$	Equation 65
6:	calculate step length $\alpha_{\ell-1}$	Equation 68
7:	$\boldsymbol{s}_\ell = \boldsymbol{s}_{\ell-1} + \alpha_{\ell-1}\boldsymbol{d}_{\ell-1}$	Equation 64
8:	**end if**	
9:	calculate cost function $j(\boldsymbol{s}_\ell)$	Algorithm 1
10:	calculate cost function's gradient $(\mathrm{d}j/\mathrm{d}\boldsymbol{s})_\ell$	Algorithm 2
11:	**if** converged **then**	Equations 66 and 67
12:	$\ell = \ell_e + 1$	
13:	**end if**	
14:	**end for**	

5 Implementation

In the forward simulation, the flow solver has to calculate the flow field for a given displacement of a boundary and return the stress on that boundary. The structural solver has to do the opposite. These functions are available in most research and commercial codes without significant modifications. Consequently, it can be stated that the partitioned forward simulation uses the flow solver and the structural solver as black boxes.

In the adjoint simulation, the exchange of operators between the solvers is avoided by communicating $\boldsymbol{C}_{f,s\Gamma}^{\mathrm{T}}\boldsymbol{a}_{f,s\Gamma}^{k}$ and $\boldsymbol{D}_{f,s\Gamma}^{\mathrm{T}}\boldsymbol{a}_{f,s\Gamma}^{n+1}$ instead of $\boldsymbol{a}_{f,s\Gamma}^{k}$ and $\boldsymbol{a}_{f,s\Gamma}^{n+1}$. At present, however, no unsteady adjoint solvers suitable for fluid-structure interaction are publicly available. Moreover, the authors are not aware of solvers which are capable of exchanging $\boldsymbol{C}_{f,s\Gamma}^{\mathrm{T}}\boldsymbol{a}_{f,s\Gamma}^{k}$ and $\boldsymbol{D}_{f,s\Gamma}^{\mathrm{T}}\boldsymbol{a}_{f,s\Gamma}^{n+1}$. Also the implementation of $\partial\boldsymbol{r}/\partial\boldsymbol{s}$ is problem dependent and would be difficult to perform with black-box solvers. So, the adjoint simulation is partitioned but does not use the solvers as black boxes.

The MATLAB source code has been written in a modular, object-oriented way. The flow solver and the structural solver are both objects which shield their internal data. They provide the required values on the fluid-structure

interface using functions. Also the extrapolation, the convergence check, the least-squares model and the cost function are objects. As a result, the layout and formatting of the main calculation routine are almost identical to Algorithm 1 and Algorithm 2.

This programming approach allows for relatively easy testing of the separate modules. It also ensures that this framework can be used for future applications. For a different problem, the flow solver, structural solver and cost function objects will have to be replaced by other objects with the same functionality. In the presented implementation, the discretisation schemes in both time and space are only first-order accurate but the presented algorithms are not limited to first-order schemes. In future work, these schemes should be replaced by higher-order schemes.

The complete code that has been used to perform the simulations is available under the Non-Profit Open Software License version 3.0 (NPOSL-3.0). This open-source approach enables verification of all presented results and full transparency of this research. All figures and tables in the results section can be generated by executing the 'Run' command.

6 Results

For the results, the tube is discretised in $m_e = 100$ segments. The parameters of the fluid-structure interaction model and the Windkessel model are listed in Table 1 and Table 2, respectively. These parameters are considered as realistic for a simple carotid artery model [49].

Figure 2 depicts the radius at the middle of the tube during one period for the minimal ($\boldsymbol{s} = -\mathbf{1}$), nominal ($\boldsymbol{s} = \mathbf{0}$) and maximal ($\boldsymbol{s} = \mathbf{1}$) values of all parameters. The differences in the other variables and at other locations are of the same order of magnitude. The cost function's gradient for $\boldsymbol{s} = \mathbf{0}$ is shown in Figure 3, with the reference in Equation 39 calculated using a sinusoidal variation of the tube's stiffness. This gradient for $\boldsymbol{s} = \mathbf{0}$ is typically used at the beginning of the parameter identification.

6.1 Verification

The adjoint calculation of the cost function's gradient is verified by comparing it to central finite differences

$$\left(\frac{\mathrm{d}j}{\mathrm{d}s_m}\right)^{fd} = \frac{j(\boldsymbol{s} + \Delta s_m) - j(\boldsymbol{s} - \Delta s_m)}{2\Delta s_m}, \tag{70}$$

with s_m and Δs_m respectively the value and the perturbation of element m in vector $\boldsymbol{s}$. This comparison is performed for different values of m and $\boldsymbol{s}$, with $\Delta s_m = 10^{-3}$ and 10^{-4}.

Table 3 lists the values from this comparison between the adjoint calculation and the finite difference calculation with $\Delta s_m = 10^{-4}$. All parameters in

the reference calculation (required for the cost function calculation in Equation 39) have been set to 1. For the forward and the adjoint calculation, the elements of the vector $\boldsymbol{s}$ have consecutively been set to -1, 0 and 1. Elements $m \in \{1, 10, 101\}$ of the gradient are analysed. The difference between both gradients is at least 5 orders of magnitude smaller than the absolute value of the gradient.

6.2 Identification

The parameters are identified for two different cases, called smooth and stepwise. In the smooth case, the reference is calculated using a sinusoidal variation of the stiffness along the tube, given by

$$s_m = 0.3 + 0.5 \sin(\pi m / m_e) \tag{71}$$

for $m = 1, \ldots, m_e$, and $s_{m_e+1} = 0.7$. In the stepwise case, the first 20 parameter values are -0.2, the next 60 are -0.6, followed by 20 times -0.3 and finally 0.1 for the parameter of the Windkessel model. In both cases, the identification starts for all parameters equal to zero. The tolerances ϵ_o and ϵ_s are both set to 10^{-6}. The number of vector pairs for the L-BFGS is limited to 15. The parameters c_1 and c_2 in the Wolfe conditions are set to 10^{-4} and 0.9, respectively [53].

Figure 4 displays the convergence of the optimisation iterations for both cases. It can easily be observed that the cost function decreases in each optimisation iteration. The convergence criterion for the first-order optimality is reached after 25 iterations (30 evaluations of the cost function and gradient) for the smooth case and after 36 iterations (42 evaluations of the cost function and gradient) for the stepwise case. During the first and second optimisation iteration, the step length deviates from 1 and the corresponding line search requires more than one evaluation of the cost function and gradient per optimisation iteration. The maximal differences in parameter value are 1.0 % and 1.2 %, respectively. Consequently, it can be concluded that both smooth and stepwise stiffness patterns can be identified.

6.3 Stability

The stability of quasi-Newton and Gauss-Seidel coupling iterations is analysed by means of the number of coupling iterations per time step. The average number of coupling iterations over 100 time steps is listed in Table 4 for different values of ρ_f and Δt. First, the reference for the cost function is calculated with a sinusoidal variation of the tube's stiffness and all subsequent calculations are then performed with $\boldsymbol{s} = \boldsymbol{0}$. The quasi-Newton iterations are applied with and without reuse of the columns from 3 time steps in the least-squares model to accelerate the convergence. The quasi-Newton iterations converge in all forward and adjoint calculations. By contrast, Gauss-Seidel iterations

only converge for a (too) large time step. A simulation is considered as unconverged if the convergence criterion is not satisfied after 25 coupling iterations in a certain time step.

In the forward simulations, the number of coupling iterations increases if the time step decreases and if the fluid density increases. This observation is consistent with prior stability analyses [24,48,25]. Here, it is demonstrated that the adjoint simulation behaves in the same way. It needs a little more coupling iterations per time step than the forward simulation, but the difference is smaller than one iteration per time step on average. Reusing columns strongly reduces the number of coupling iterations per time step. For both the forward and the adjoint equations, Gauss-Seidel iterations are unstable in most cases.

Figure 5 shows the evolution of the coupling residual $||\boldsymbol{r}_{s\Gamma}^k||_2$ during the first 20 coupling iterations of the forward and adjoint simulations with $\rho_f = 1060\,\mathrm{kg/m^3}$ and $\Delta s = 0.01\,\mathrm{s}$. The Gauss-Seidel iterations diverge with a constant slope in the forward simulation. As this forward simulation does not finish successfully, the corresponding adjoint simulation cannot be started. Conversely, the first time step of the forward simulation converges after 6 quasi-Newton iterations. In this first time step, there is as expected no difference between with and without reuse of columns from previous time steps. The coupling residual jumps up in iteration 7, which is the beginning of the second time step. From that coupling iteration on, the convergence is faster with reuse than without.

Figure 6 depicts the columns of the matrix $\boldsymbol{V}^k$ for $k = 5$ in the first time step of a typical forward and adjoint simulation. It can be observed that the wave number of all columns is low for the forward simulation, as explained in [48,25,54]. Moreover, the columns in the adjoint simulation also have a relatively low wave number. This suggests that the stability of the coupling iterations is similar for the forward and the adjoint simulation. Consequently, the coupling techniques which have been developed for the partitioned simulation of fluid-structure interaction problems can also be applied to the partitioned simulation of the corresponding adjoint problems.

7 Conclusions

In this work, the stiffness of each segment of a tube and of the Windkessel model at the tube's outlet are identified using gradient-based optimisation. The cost function is calculated with an unsteady fluid-structure interaction simulation; the gradient is obtained from an unsteady adjoint simulation. Both the forward and the adjoint simulation are partitioned with a quasi-Newton coupling algorithm (IQN-ILS). The stability of the coupling iterations is similar in the forward and the adjoint problem. Since the identification is performed using optimisation, the presented algorithms are applicable to the optimisation of unsteady fluid-structure interaction in general.

The forward calculation only requires functionality of the flow solver and structural solver that is readily available in solvers suitable for fluid-structure

interaction, without any modification of the codes. Consequently, the forward calculation uses both solvers as black boxes. For the unsteady adjoint calculation, however, no suitable open-source or commercial solvers are publicly available. Currently, only steady adjoint solvers are publicly available. In this study, only variables on the fluid-structure interface are exchanged in the adjoint simulation and it is shown which functions in the adjoint flow solver and structural solver are required. Nevertheless, calculating the derivatives of the governing equations with respect to the parameters will always be difficult to implement in a black-box solver, except when using finite differences.

Finally, quasi-Newton algorithms are used for the optimisation iterations and the coupling iterations in the forward and adjoint calculations. Both algorithms approximate the inverse of a matrix to avoid the solution of a dense linear system and update this approximation in every iteration. Moreover, the product of these matrices with a vector is calculated without explicitly constructing them. Obviously, this is not strictly necessary for this demonstration example. However, together with the modular approach, it will allow to use the developed framework for future broader applications.

Acknowledgements Joris Degroote gratefully acknowledges funding by a post-doctoral fellowship of the Research Foundation - Flanders (FWO).

References

1. A. Quarteroni, M. Tuveri, A. Veneziani, Computing and Visualization in Science **2**(4), 163 (2000)
2. L. Formaggia, D. Lamponi, A. Quarteroni, Journal of Engineering Mathematics **47**(3–4), 251 (2003)
3. M. Bonnet, A. Constantinescu, Inverse Problems **21**(2), R1 (2005)
4. C. Bertoglio, P. Moireau, J.F. Gerbeau, RR INRIA **7657**, 1 (2011)
5. D. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning* (Addison Wesley, 1989)
6. R. Eberhart, Y. Shi, J. Kennedy, *Swarm Intelligence* (Morgan Kaufmann, 2001)
7. S. Kirkpatrick, C. Gelatt, M. Vecchi, Science **220**(4598), 671 (1983)
8. D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, K. Crombecq, Journal of Machine Learning Research **11**, 2051 (2010)
9. A. Stück, F. Camelli, R. Löhner, International Journal for Numerical Methods in Fluids **64**(4), 443 (2010)
10. R. Balasubramanian, J. Newman, International Journal for Numerical Methods in Engineering **66**(2), 297 (2006)
11. D. Papadimitriou, K. Giannakoglou, International Journal for Numerical Methods in Fluids **56**(10), 1929 (2008)
12. M. Heil, Computer Methods in Applied Mechanics and Engineering **193**(1–2), 1 (2004)
13. B. Hübner, E. Walhorn, D. Dinkler, Computer Methods in Applied Mechanics and Engineering **193**(23–26), 2087 (2004)
14. M. Gee, U. Küttler, W. Wall, International Journal for Numerical Methods in Engineering **85**(8), 987 (2011)
15. C. Felippa, K. Park, C. Farhat, Computer Methods in Applied Mechanics and Engineering **190**(24–25), 3247 (2001)
16. C. Farhat, K. van der Zee, P. Geuzaine, Computer Methods in Applied Mechanics and Engineering **195**(17–18), 1973 (2006)
17. E. van Brummelen, Journal of Applied Mechanics **76**(2), 021206 (2009)

18. D. Mok, W. Wall, E. Ramm, in *Computational Fluid and Solid Mechanics*, ed. by K.J. Bathe (Elsevier, 2001), pp. 1325–1328
19. R. Wüchner, A. Kupzok, K.U. Bletzinger, International Journal for Numerical Methods in Fluids **54**(6–8), 945 (2007)
20. U. Küttler, W. Wall, Computational Mechanics **43**(1), 61 (2008)
21. C. Michler, E. van Brummelen, R. de Borst, International Journal for Numerical Methods in Fluids **47**(10-11), 1189 (2005)
22. J. Degroote, K.J. Bathe, J. Vierendeels, Computers & Structures **87**(11–12), 793 (2009)
23. M. Hojjat, E. Stavropoulou, T. Gallinger, U. Israel, R. Wüchner, K.U. Bletzinger, in *Fluid-Structure Interaction II: Modelling, Simulation, Optimization*, ed. by H.J. Bungartz, M. Mehl, M. Schäfer, Lecture Notes in Computational Science and Engineering (Springer, Berlin Heidelberg, 2010), pp. 351–381
24. P. Causin, J.F. Gerbeau, F. Nobile, Computer Methods in Applied Mechanics and Engineering **194**(42–44), 4506 (2005)
25. J. Degroote, S. Annerel, J. Vierendeels, Computers & Structures **88**(5–6), 263 (2010)
26. J. Degroote, R. Haelterman, S. Annerel, P. Bruggeman, J. Vierendeels, Computers & Structures **88**(7–8), 446 (2010)
27. J. Martins, J. Alonso, J. Reuther, Journal of Aircraft **41**(3), 523 (2004)
28. J. Martins, J. Alonso, J. Reuther, Optimization and Engineering **6**(1), 33 (2005)
29. K. Maute, M. Nikbay, C. Farhat, American Institute of Aeronautics and Astronautics Journal **39**(11), 2051 (2001)
30. K. Maute, M. Nikbay, C. Farhat, International Journal for Numerical Methods in Engineering **56**(6), 911 (2003)
31. A. Fazzolari, N. Gauger, J. Brezillon, Journal of Computational and Applied Mathematics **203**(2), 548 (2007)
32. N. Gauger, A. Fazzolari, *MegaDesign and MegaOpt - German Initiatives for Aerodynamic Simulation and Optimization in Aircraft Design, Results of the closing symposium of the MegaDesign and MegaOpt projects, Braunschweig, Germany, 23 - 24 May, 2007* (Springer-Verlag, Berlin Heidelberg, 2009), *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, vol. 107, chap. Adjoint Methods for Coupled CFD-CSM Optimization, pp. 237–246
33. K. Palaniappan, P. Sahu, J. Alonso, A. Jameson, in *44^{th} AIAA Aerospace Sciences Meeting and Exhibit* (Reno, NV, USA, 2006), pp. 1–11. AIAA 2006-844
34. K. Palaniappan, P. Sahu, J. Alonso, A. Jameson, in *47^{th} AIAA Aerospace Sciences Meeting* (Orlando, FL, USA, 2009), pp. 1–20. AIAA 2009-148
35. M. Rumpfkeil, D. Zingg, Optimization and Engineering **11**(1), 5 (2010)
36. E. Nielsen, B. Diskin, N. Yamaleev, in *19^{th} AIAA Computational Fluid Dynamics Conference* (San Antonio, TX, USA, 2009), pp. 1–22. AIAA 2009-3802
37. A. Griewank, A. Walther, ACM Transactions on Mathematical Software **26**(1), 19 (2000)
38. J. Sternberg, A. Griewank, *Automatic Differentiation: Applications, Theory, and Implementations* (Springer-Verlag, Berlin Heidelberg, 2006), *Lecture Notes in Computational Science and Engineering*, vol. 50, chap. Reduction of Storage Requirement by Checkpointing for Time-Dependent Optimal Control Problems in ODEs, pp. 99–110
39. Q. Wang, P. Moin, G. Iaccarino, SIAM Journal on Scientific Computing **31**(4), 2549 (2009)
40. P.Y. Lagrée, The European Physical Journal - Applied Physics **9**(2), 153 (2000)
41. V. Martin, F. Clément, A. Decoene, J.F. Gerbeau, ESAIM: Proceedings **14**, 174 (2005)
42. J.F. Gerbeau, M. Vidrascu, ESAIM: Mathematical Modelling and Numerical Analysis **37**(4), 631 (2003)
43. G. Cowper, Journal of Applied Mechanics **33**(2), 335 (1966)
44. G. Kennedy, J. Hansen, J. Martins, International Journal of Solids and Structures **48**(16–17), 2373 (2011)
45. J. Vierendeels, K. Riemslagh, E. Dick, Journal of Computational Physics **154**(2), 310 (1999)
46. J. Vierendeels, K. Dumont, E. Dick, P. Verdonck, AIAA Journal **43**(12), 2549 (2005)
47. F. Li, Z. Sun, Journal of Computational and Applied Mathematics **200**(2), 606 (2007)
48. J. Degroote, P. Bruggeman, R. Haelterman, J. Vierendeels, Computers & Structures **86**(23–24), 2224 (2008)

49. I. Vignon-Clementel, C. Figueroa, K. Jansen, C. Taylor, Computer Methods in Biomechanics and Biomedical Engineering **13**(5), 625 (2010)
50. R. Haelterman, J. Degroote, D. Van Heule, J. Vierendeels, SIAM Journal on Numerical Analysis **47**(3), 2347 (2009)
51. R. Haelterman, J. Degroote, D. Van Heule, J. Vierendeels, SIAM Journal on Numerical Analysis **47**(6), 4660 (2010)
52. A. Stück, F. Camelli, R. Löhner, in *48th AIAA Aerospace Sciences Meeting* (Orlando, FL, USA, 2010), pp. 1–30. AIAA 2010-1430
53. J. Nocedal, S. Wright, *Numerical Optimization*. Springer Series in Operations Research (Springer, 1999)
54. E. van Brummelen, International Journal for Numerical Methods in Fluids **65**(1–3), 3 (2011)

Figures

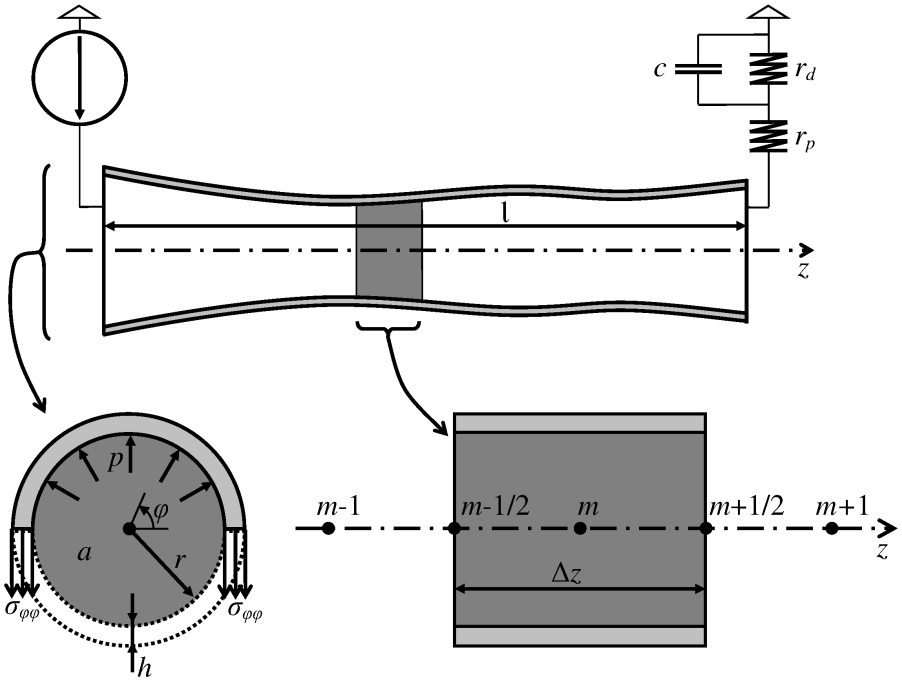


Fig. 1: The model for blood flow in an artery with details of the cross-section and a control volume used in the discretisation of the governing equations. Also the prescribed velocity at the inlet and the Windkessel model at the outlet are depicted.

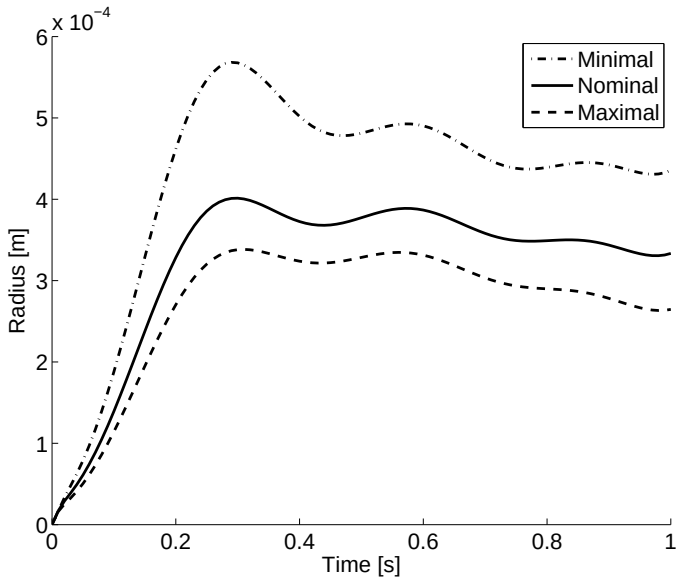


Fig. 2: The radius linearised with respect to r_o at the middle of the tube ($z = \ell/2$) as a function of time for minimal ($\boldsymbol{s} = -\mathbf{1}$), nominal ($\boldsymbol{s} = \mathbf{0}$) and maximal values ($\boldsymbol{s} = \mathbf{1}$) of all parameters.

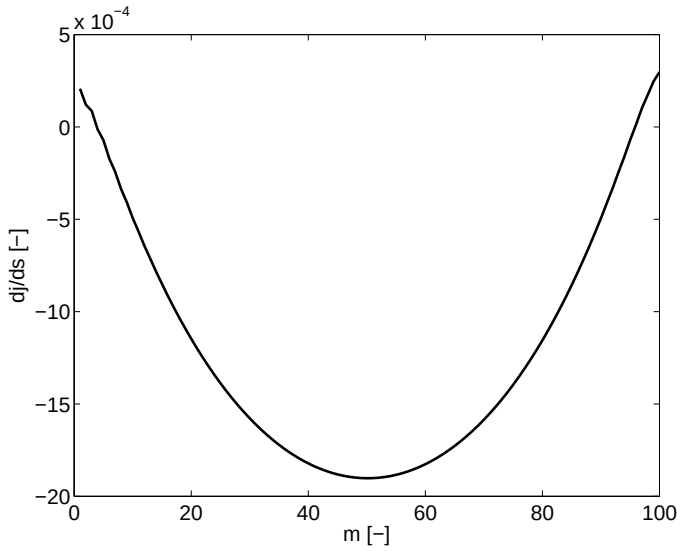


Fig. 3: The cost function's gradient with respect to the stiffness of the tube segments for $\boldsymbol{s} = \mathbf{0}$ if the reference is calculated with a sinusoidal variation of the stiffness.

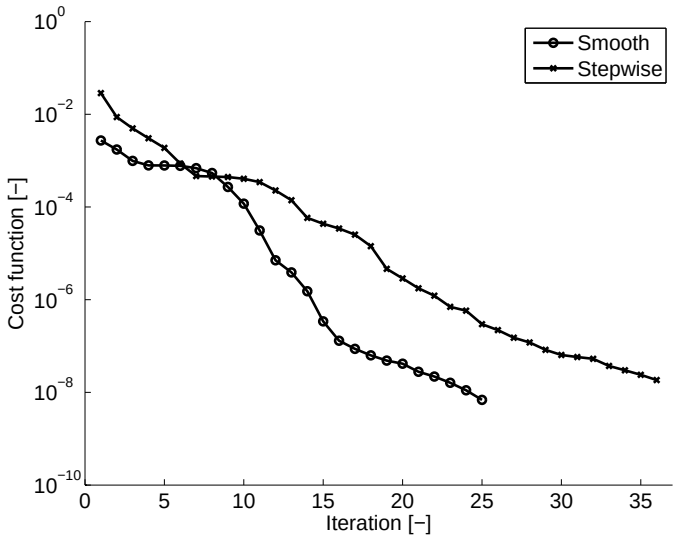
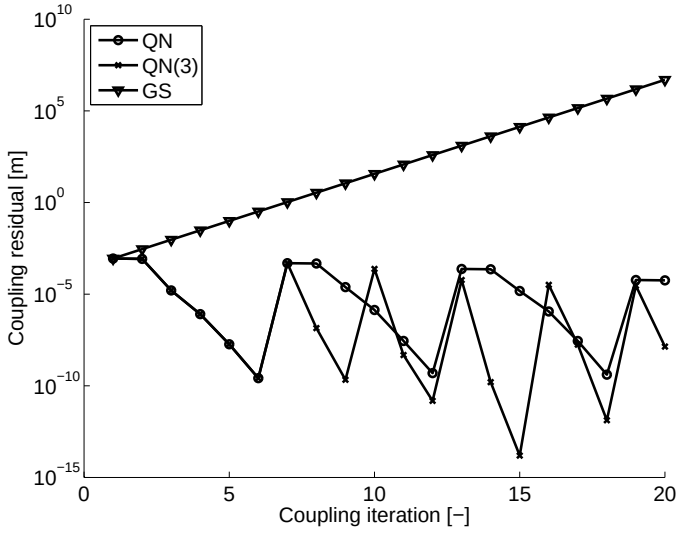
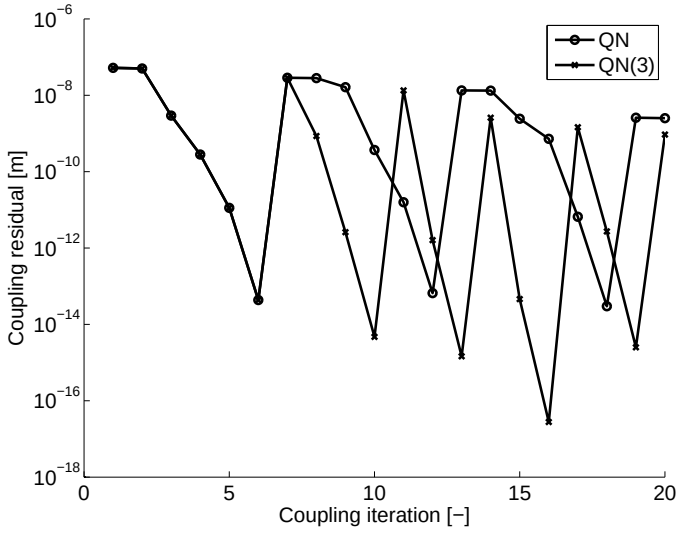


Fig. 4: The convergence of the optimisation iterations for the smooth and stepwise case of the parameter identification.

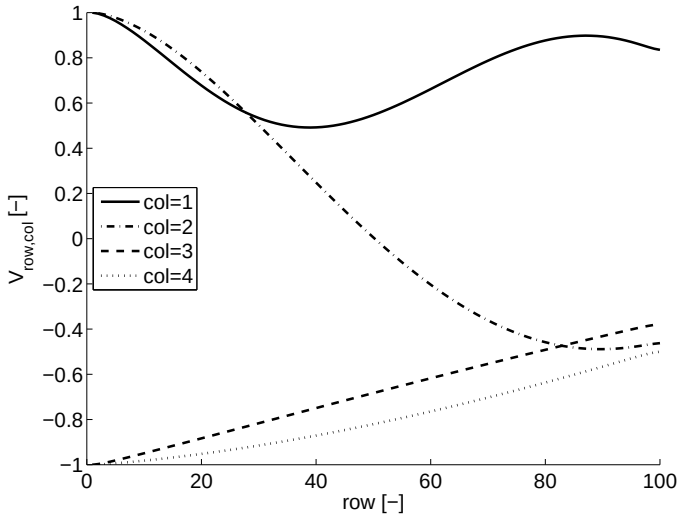


(a)

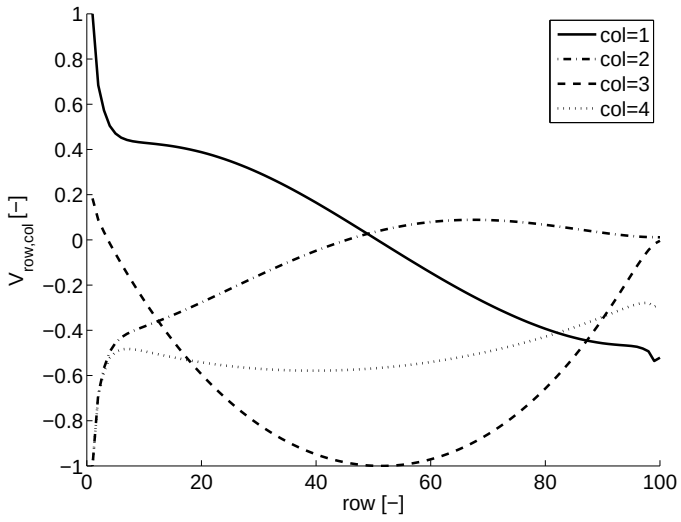


(b)

Fig. 5: The convergence of the first 20 coupling iterations in the (a) forward and (b) adjoint simulation with $\rho_f = 1060\,\mathrm{kg/m^3}$ and $\Delta s = 0.01\,\mathrm{s}$. The Gauss-Seidel iterations do not converge in the forward calculation, so the corresponding adjoint calculation cannot be performed. The quasi-Newton iterations converge quickly and jump up at the beginning of each new time step.



(a)



(b)

Fig. 6: The normalised columns of the matrix $\boldsymbol{V}^5$ in the first time step of a typical (a) forward and (b) adjoint simulation. The normalisation of each column is performed by dividing it by its maximal absolute value.

Tables

Table 1: The parameters of the fluid-structure interaction model [49].

ρ_f	$1060\,\mathrm{kg/m^3}$	ρ_s	$1000\,\mathrm{kg/m^3}$
r_o	$3{\cdot}10^{-3}\,\mathrm{m}$	E_o	$4{\cdot}10^{5}\,\mathrm{Pa}$
h	$3{\cdot}10^{-4}\,\mathrm{m}$	G	$4{\cdot}10^{5}\,\mathrm{Pa}$
ℓ	$0.126\,\mathrm{m}$	ν	0.5

Table 2: The parameters of the Windkessel model [49].

c_o	$6.35{\cdot}10^{-10}\,\mathrm{m}^3/\mathrm{Pa}$	0.0846 ml/mmHg
r_p	$2.834{\cdot}10^{8}\,\mathrm{Pa{\cdot}s/m}^3$	2.126 mmHg·s/ml
r_d	$1.768{\cdot}10^{9}\,\mathrm{Pa{\cdot}s/m}^3$	13.263 mmHg·s/ml
t_b	1 s	

Table 3: The verification of the gradient calculation by means of a comparison with finite differences using step size $\Delta s_m = 10^{-4}$.

m	$\boldsymbol{s}$	$\mathrm{d}j/\mathrm{d}s_m$	$(\mathrm{d}j/\mathrm{d}s_m)^{fd}$	$\lvert\mathrm{d}j/\mathrm{d}s_m - (\mathrm{d}j/\mathrm{d}s_m)^{fd}\rvert$
1	-1	-9.4184248e-03	-9.4184185e-03	6.3051349e-09
1	0	-1.2726096e-03	-1.2726133e-03	3.6851994e-09
1	1	0.0000000e+00	-1.0685503e-12	1.0685503e-12
10	-1	-1.0022075e-02	-1.0022066e-02	9.3869615e-09
10	0	-1.3523483e-03	-1.3523465e-03	1.8668861e-09
10	1	0.0000000e+00	-4.2920656e-13	4.2920656e-13
101	-1	4.8240110e-01	4.8240119e-01	8.5818958e-08
101	0	7.0555794e-02	7.0555806e-02	1.1679346e-08
101	1	0.0000000e+00	-4.8473439e-11	4.8473439e-11

Table 4: The average number of coupling iterations per time step in the forward (adjoint) simulation as a function of the time step Δt and the fluid density ρ_f. The top, middle and bottom of the table respectively list the number of quasi-Newton iterations without reuse (QN), quasi-Newton iterations with reuse of columns from 3 time steps (QN(3)) and Gauss-Seidel (GS) iterations.

	$\rho_f \downarrow$	$\Delta t \rightarrow$	10^{-1}	10^{-2}	10^{-3}
QN	106		3.50 (4.01)	4.09 (5.02)	7.10 (7.81)
	1060		3.99 (4.00)	5.27 (6.00)	10.62 (11.17)
	10600		4.21 (5.00)	7.16 (7.25)	16.44 (17.42)
	$\rho_f \downarrow$	$\Delta t \rightarrow$	10^{-1}	10^{-2}	10^{-3}
QN(3)	106		3.00 (3.05)	3.02 (3.07)	3.17 (3.28)
	1060		3.01 (3.01)	3.03 (3.06)	3.77 (4.30)
	10600		3.01 (3.02)	3.13 (3.22)	6.46 (6.48)
	$\rho_f \downarrow$	$\Delta t \rightarrow$	10^{-1}	10^{-2}	10^{-3}
GS	106		11.00 (10.97)	— (—)	— (—)
	1060		11.00 (11.00)	— (—)	— (—)
	10600		14.40 (14.22)	— (—)	— (—)