Solver-agnostic multi-fidelity coupling framework for the partitioned simulation of fluid-structure interactions

Carlos Fernando Baptista

Master of Science Thesis

Aerodynamics

Faculty of Aerospace Engineering

TU Delft

Delft University of Technology

Solver-agnostic multi-fidelity coupling framework for the partitioned simulation of fluid-structure interactions

Master of Science Thesis

For the degree of Master of Science in Aerospace Engineering at
Delft University of Technology

Carlos Fernando Baptista

August 12, 2015

Faculty of Aerospace Engineering · Delft University of Technology

The work in this thesis was performed under the banner of the FP7 project “Future Fast Aeroelastic Simulation Technologies”. The project is hereby gratefully acknowledged.

Delft University of Technology

Delft University of Technology
Faculty of Aerospace Engineering
Department of
Aerodynamics, Wind Energy, Flight Performance and Propulsion

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled

Solver-agnostic multi-fidelity coupling framework for the partitioned simulation of fluid-structure interactions

by

Carlos Fernando Baptista

in fulfillment of the requirements for the degree of

Master of Science in Aerospace Engineering

Dated: August 12, 2015

Supervisor(s):

Dr. ir. Alexander Herman van Zuijlen

Ir. Thomas Peter Scholcz

Reader(s):

Dr. ir. Carlos Simao Ferreira

Dr. ir. Roeland de Breuker

Preface

When still at high school, three years before going to university, I already had decided I wanted to study Aerospace Engineering. I was a bit anxious though because Aerospace Engineering is considered one of the most difficult programmes here in Delft. Back then I did not know whether my proficiency in math and physics was sufficient for such a tough programme. But I always love a challenge. Today marks a great day as I am delivering the finishing touches on my thesis for the Master of Science programme in Aerospace Engineering.

For my thesis I developed a reusable solver-agnostic coupling framework to accelerate the future set up of partitioned fluid-structure interactions. The framework was developed with extensibility in mind to provide a basis for the future implementation and assessment of new coupling algorithms. This framework is baptised "CASMIR" (Coupling Algorithms for Strongly-coupled Multi-physics Interaction Research).

I would like to pay special thanks to my supervisors Alexander van Zuijlen and Thomas Scholcz. I thank you, Thomas, for bringing me up to speed with space mapping and for providing me with several pieces of code to help me along the way. I thank you, Alexander, for your guidance and for aiding me with the design and debugging of CASMIR.

I would also like to thank my parents, Fernando and Isilda and my brother Roque for all their moral support and motivational speeches. Especially my parents, thanks for the speeches.

I would like to thank my best friend Desiree de Vreede for all the fun days and nights we spend working together on our separate theses.

Finally, I would like to thank my girlfriend Symphony Veira for having my back the entire time. You are my "gnoe".

Carlos Fernando Baptista,
Delft, August 2015

Abstract

Fluid-structure interactions (FSI) are multi-physical phenomena where the dynamics of a fluid flow and the dynamics of a moving/deforming structure influence one another simultaneously. The accurate modelling, simulation and analysis of FSI is crucial for many engineering applications. However, high-fidelity FSI simulations are currently too computationally expensive for industrial purposes. The industry is in need of more efficient software to perform accurate FSI analyses at reduced computational cost.

The complexity of developing such software is acknowledged and accounted for by preserving software modularity. Using black-box mono-physics solvers in a partitioned framework is one step towards ensuring software modularity. Partitioned procedures require a coupling algorithm to iteratively reduce errors related to the partitioning of the physical domains. Implementing coupling algorithms directly into each solver results in duplicitous work. Instead all algorithms related to coupling procedures should be centralised into a single unit. To this end a solver-agnostic framework is developed for the partitioned simulation of strongly-coupled fluid-structure interactions. This framework is named CASMIR (Coupling Algorithms for Strongly-Coupled Multi-physics Interaction Research).

By formulating the coupling of a partitioned FSI problem as a root-finding problem, the partitioning error can be reduced by employing Newton's method. Because the numerical solvers for the fluid dynamics and the structure dynamics are used as black boxes (inverse-)Jacobians are not readily available. The task of a coupling algorithm therefore boils down to approximating (inverse-)Jacobians. The coupling algorithms are, hence, quasi-Newton method. CASMIR currently incorporates Gauss-Seidel, Aitken Δ^2 underrelaxation, Broyden and IQN-ILS(r) as coupling algorithms for mono-fidelity partitioned simulations.

Each coupling iteration involves the evaluation of both the fluid and the structure solver. As multiple coupling iterations are needed per time step for convergence highly efficient coupling algorithms are needed to obtain highly accurate results while maintaining the computational cost low. To this end, space mapping is introduced. Space mapping is a multi-fidelity tool that accelerates the iterative convergence process of a expensive high-fidelity model by exploiting a cheap low-fidelity model. Many variants of space mapping exist, however, only Aggressive Space Mapping (ASM) is currently available in CASMIR.

ASM is basically a multi-fidelity extension of a mono-fidelity root-finding problem. The aforementioned coupling algorithms can therefore be fitted in the framework of ASM to, once again, approximate (inverse-)Jacobians. Hence, the coupling algorithms for multi-fidelity

partitioned simulations implemented in CASMIR are developed by incorporating the mono-fidelity coupling algoritms into the framework of ASM.

The performance of the mono-fidelity and multi-fidelity coupling algorithms were assessed on two test problems, namely, the 2D supersonic panel flutter problem and the quasi-one-dimensional channel flow problem. IQN-ILS(r) was found to be the most efficient mono-fidelity coupling algorithm. IQN-ILS(r) is especially efficient when it reuses data from previous time steps (i.e. $r > 0$)to provide more accurate (inverse-)Jacobian approximations.

Computational speedups relative to IQN-ILS(0) (i.e. no reuse) were obtained by the multi-fidelity coupling algorithms that combined either Gauss-Seidel, Aitken or Broyden with Aggressive Space Mapping. The combination of IQN-ILS(r) and Aggressive Space Mapping performed very poorly. It is believed that the implementation is faulty and not that the particular combination is an inefficient one. Further debugging is needed.

Although, the multi-fidelity coupling algorithms that do work properly achieved speedups relative to IQN-ILS(0), they are however less efficient than IQN-ILS(r) for $r > 0$.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project introduction

Fluid-structure interactions (FSI) are multi-physical phenomena where the dynamics of a fluid flow and the dynamics of a moving/deforming structure influence one another simultaneously. The modelling, simulation and analysis of FSI is crucial for the design of many fluid-structure systems. Examples include wind-induced vibrations in bridges and high-rise buildings, conversion of kinetic energy in wind to electricity by wind turbines, blood flow through veigns and mechanical heart-valves and flutter of aircraft wings.

Simulations of fluid-structure systems using high-fidelity models are needed to not only assess the performance of the system accurately, yet also to thorougly study instabilities due to FSI. High-fidelity FSI simulations of strongly-coupled fluid-structure systems are very computationally expensive and this limits their application in industry. Therefore, the aerospace industry (among others) is in need of more efficient computational algorithms. FFAST (**F**uture **F**ast **A**eroelastic **S**imulation **T**echnologies), an European project set to address this industrial demand is introduced hereafter, followed by recent developments at the Delft University of Technology, in accordance with the FFAST project.

1.1.1 FFAST: Future Fast Aeroelastic Simulation Technologies

Unsteady loads play a crucial role in the design and development of aircrafts and have a large impact on the detailed structural design, aerodynamic properties and flight performance. Load types include, among others, dynamic gust loads, manoeuvre responses and landing impacts. The internal stress distribution that an aircraft experiences in service depends largely on the load conditions (combination of flight configurations and performed manoeuvres). Unfortunately, the load conditions that generate the largest internal stresses are not known a priori. To estimate the maximum stresses a batch of aeroelastic simulations is required to cover all possible load conditions. A complete load calculation cycle for a civil aircraft takes more than 6 weeks and has to be repeated after each structural design iteration. High-fidelity aeroelastic simulations are therefore prohibitive.

FFAST is an European project within FP7 (Seventh Framework Programme). The objectives are to develop, implement and assess simulation technologies to accelerate future aircraft design. This is to be achieved by improving both the efficiency and accuracy of aeroelastic load prediction procedures using unique critical load identification methods and reduced order modelling. Current low-fidelity models are to be replaced by more accurate aeroelastic simulations whilst not increasing the overall computational costs for the prediction of unsteady loads.

Research conducted within the FFAST project pertain to three major areas, being

- Faster identification of critial load cases,
- Extraction of aerodynamic and aeroelastic reduced order models (ROM),
- Reduced-order model acceleration of full-order model.

These objectives are pursued by a transnational consortium consisting of

- Delft University of Technology (DUT)
- Politecnico di Milano
- University of Bristol
- University of Cape Town (UCT)
- University of Liverpool
- Council of Scientitifc and Industrial Research (CSIR)
- Deutsches Zentrum f ur Luft- und Raumfahrt (DLR)
- Institut National de Recherche en Informatique et en Automatique (INRIA)
- Institute for Information Transmission Problems (IITP)
- International Research Institute for Advanced Systems (IRIAS)
- Airbus
- European Aeronautic Defence and Space company (EADS-MS)
- Numeca International
- Optimad Engineering

1.1.2 Recent developments at the Delft University of Technology

Research at the faculty of Aerospace Engineering at the Delft University of Technology is geared towards the acceleration of high-fidelity aeroelastic simulations using aeroelastic reduced order models. The multidisciplinary complexity of the research is acknowledged and accounted for by preserving software modularity through incorporating black-box solvers. This allows the associated engineering communities to independently implement new code and to reuse existing software.

Recent developments related to the FFAST project at the Delft University of Technology include the doctoral research of Ir. T. P. Scholcz [1, 2] and the graduating research of Ir. L. Florentie [3].

Scholcz has looked into multi-fidelity techniques to accelerate a computationally expensive, yet accurate, fine model by exploiting a computationally cheap, yet inaccurate, coarse model. One technique is space-mapping, a multi-fidelity optimisation tool developed by Bandler et. al in 1994 [4]. Space-mapping defines a mapping function that matches a fine model response to a coarse model response. The core principle of space-mapping is to iteratively

determine the inverse mapping function. An optimised fine model solution is then obtained by applying the inverse mapping function on the optimised coarse model solution. For a reduction in computational cost it must hold that the evaluation of the coarse model is considerbly cheaper than evaluating the fine model.

Florentie assessed the relative performance of several space-mapping variants applied to strongly-coupled fluid-structure interaction (FSI) simulations. Among others, especially the Aggressive Space Mapping (ASM) variant performed considerably well. ASM is essentially a reformulation of a quasi-Newton method and therefore needs an algorithm to approximate Jacobians. The original formulation of ASM [5] used Broyden's method [6] for the approximation. Broyden's approach is to use an initial guess for the full Jacobian, followed by rank-one updates for subsequent Newton iterations.

1.1.3 Project outline

In the FSI community, however, more robust Jacobian approximation algorithms have been developed. IQN-ILS (**I**nterface **Q**uasi-**N**ewton with **I**nverse-Jacobian from **L**east **S**quares) is the current state-of-the-art and uses a multi-point least-squares formulation to approximate an inverse-Jacobian based soleley on quantities defined on the fluid-structure interface. Superiority of IQN-ILS in terms of efficiency follows from reusing information of multiple Newton iterations, multiple time steps, approximating directly the inverse-Jacobian instead of the Jacobian (thus saving one system solve) and the restriction of its operations to only the fluid-structure interface.

In this project, a reusable solver-agnostic quasi-Newton-based coupling framework for the partitioned simulation of strongly-coupled fluid-structure interactions is to be developed. Solver-agnosticism implies that the framework is developed completely independent from external solvers to allow pure black-box functionalities and thus preserving software modularity. The framework baptised CASMIR, will incorporate several mono-fidelity coupling algorithms (see Chapter 3) as well as several multi-fidelity coupling algorithms (see Chapter 4). The state-of-the-art is to be the unision of ASM and IQN-ILS. The IQN-ILS algorithm is to be integrated into ASM, replacing the original and less efficient Broyden's method. Finally the performance of all implemented algorithms is to be assesed using test cases defined in Chapter 6.

1.2 Introduction to fluid-structure interaction simulations

The simulation of fluid-structure interactions combines the expertise of both the Computational Fluid Dynamics (CFD) and the Computational Structure Dynamics (CSD) communities. The governing equations used to model fluid flow are usually developed independently from the governing equations used to model the dynamics of a flexible structure. The physically-distinct domains are coupled by enforcing boundary conditions at the fluid-structure interface. This section introduces the relevant boundary conditions. Approaches for simulating fluid-structure interactions are discussed subsequently. The section closes with a succinct overview of the procedures required to couple the physically-distinct domains both in space and in time.

1.2.1 Boundary conditions

To enable interactions between the two physically-distinct domains Figure 1.1, the equations that govern the fluid flow are coupled with the equations that govern the flexible structure by enforcing two boundary conditions.

The first condition is a kinematic boundary condition stated as

$$\mathbf{u} = \dot{\mathbf{x}} \qquad \text{on } \Gamma_I, \tag{1.1}$$

where $\mathbf{u}$ is the velocity field of the flow, $\dot{\mathbf{x}}$ is the rate of displacement of the structure and Γ_I denotes the fluid-structure interface. The kinematic boundary condition simply implies that, at the interface, the normal and tangential components of the flow field are equal to the corresponding velocity components of the structure. In the fluid dynamics community this condition is also known as the no-slip condition for viscous flows. In inviscid flows, however, only the normal components should be equal.

The second condition is a dynamic boundary condition stated as

$$\mathbf{n}_f \cdot \mathbf{T}_f = -\mathbf{n}_s \cdot \mathbf{T}_s \qquad \text{on } \Gamma_I, \tag{1.2}$$

where $\mathbf{n}_f$ is the vector normal to the fluid-side of the fluid-structure interface, $\mathbf{T}_f$ is the fluid stress tensor acting on the fluid-side of the fluid-structure interface, $\mathbf{n}_s$ and $\mathbf{T}_s$ are analogously defined for the structure-side of the fluid-structure interface. The dynamic boundary condition is simply Newton's third law of motion (action is minus reaction), hence it relates the stresses acting normal and tangential to the interface to be equal in magnitude yet opposite in sign.

1.2.2 Coupling approaches

Solvers developed by the CFD and CSD communities have flourished for several decades. These solvers have mainly been specialised to handle either fluid dynamics or structure dynamics quite well with little or no support for the other physical domain. Therefore, for the simulation of fluid-structure interactions a choice has to be made whether to reuse existing software as is, extend existing software or develop completely new software.

Two prevalent approaches exist for FSI simulations. The two sets of governing equations can be solved simultaneously in a unified framework (i.e. monolithic approach) or separately in a sequential framework (i.e. partitioned approach).

Monolithic coupling approach

In the monolithic approach (see Figure 1.1) a single solver $\mathcal{M}$ is used to solve a single large set of algebraic equations at each time level, containing flow equations, structure equations and boundary conditions. This approach is favoured in terms of stability and accuracy, yet is difficult to maintain due to a lack of software modularity. Extending a monolithic solver is therefore difficult and usually a new dedicated solver is build for each new type of coupling problem that is governed by previously unconsidered physics [7, 8, 9, 10, 11].

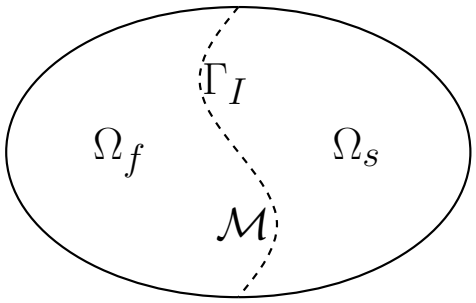


Figure 1.1: Idealised representation of the monolithic coupling approach

Partitioned coupling approach

A different approach is to solve the flow equations and the structure equations separately using two distinct solvers (see Figure 1.2). Starting from an initial guess, the governing equations of each physical domain are then solved sequentially. Partitioning of the physical domains therefore has an asynchronicity of state variables as direct consequence. This introduces an additional error: the partitioning error. A coupling algorithm is then required to reduce the partitioning error [8, 9] per time level by repeatedly invoking the distinct solvers sequentially . This process is called sub-iterating and it results in multiple system solves (i.e. coupling-iterations) per time level.

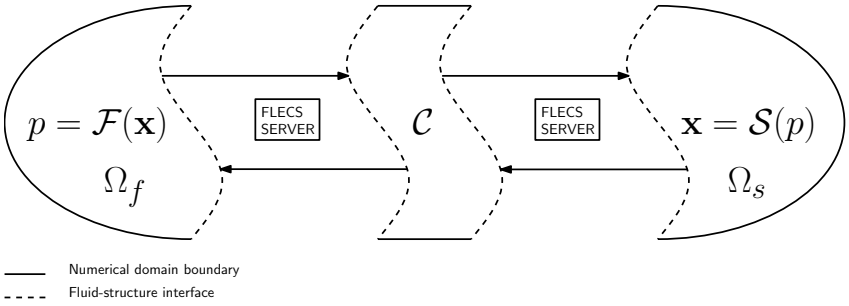


Figure 1.2: Representation of the partitioned coupling approach

Defining $\mathcal{F}$ as the flow solver operator taking the structure displacement, $\mathbf{d}$, as input and producing the fluid pressure, p, as output and defining similarly the structure solver operator $\mathcal{S}$ taking and producing the reverse, a residual operator is then defined as

$$\mathbf{r} = \mathcal{R}(\mathbf{d}) \equiv \mathcal{S} \circ \mathcal{F}(\mathbf{d}) - \mathbf{d}. \tag{1.3}$$

The purpose of the coupling algorithm, $\mathcal{C}$, is to reduce the residual described by equation (1.3) by sequentially solving the flow equations and the structure equations multiple times at each time level, until convergence of the residual up to a predefined tolerance. The interaction intensity of a FSI problem depends largely on the compressbility of the fluid and the flexibility of the structure. When a single coupling-iteration per time level reduces the residual sufficiently, the FSI problem is called weakly- or loosely-coupled [12, 13], otherwise the FSI problem is called strongly-coupled [13].

The need for multiple coupling-iterations per time level in a strongly-coupled problem increases the amount of computational work considerably, hence the monolithic approach is more efficient computational-wise. On the other hand, in the partitoned approach, existing solvers can be reused that were specifically optimised for a specific physical domain [7, 14, 15]. This is generally preferred due to the confidence in reliable mature software. In addition, the solvers are easily interchangeable as long as the flow and the structure solvers are considered black boxes [7]. This way a wide body of aeroelastic/FSI problems can be solved accurately by selecting optimised solvers for each specific domain and problem.

1.2.3 Partitioned coupling

Partitioned simulations of fluid-structure interactions consist of three fundamental pillars:

1. Time stepping
2. Sub-iterating
3. Interpolating

Solvers employed by the CFD and the CSD communities divide time into a number of discrete time levels and solve for the state of the dynamical system at those time levels. This procedure is known as time stepping.

If a FSI problem is considered to be strongly-coupled, a sub-iterating procedure has to be applied. As coupling algorithm either fixed-point iterations or quasi-Newton methods can be used.

Finally, if the physically-distinct domains are discretised using different meshes, overlaps and/or gaps can exist at the fluid-structure interface. Interpolation is then required to correctly transfer physical quantities across the interface.

These three fundamental pillars are discussed in greater detail in Chapter 2.

1.3 Research objective and plan

Numerical solvers for simulating fluid dynamics or structure dynamics have been developed for several decades. These codes were initially written specifically for a single physical domain. Modern day engineering design, however, requires software capable of performing multi-physical simulations. Software modularity is crucial in order to cope with the increasing complexity of software developed by the CFD and CSD communities.

Using black-box mono-physics solvers in a partitioned framework is one step towards ensuring software modularity. However, implementing coupling algorithms directly into each solver results in duplicitous work. Instead all algorithms related to coupling procedures should be centralised into a single unit: CASMIR, the coupling framework.

The goals for the initial version of CASMIR are to uphold several design properties and incorporate a basic set of functionality. The desired design properties are:

- solver agnostic
- non-intrusive

- user friendly
- extensible

Furthermore it is desired for CASMIR to be capable of:

- Handling data transfer between independently developed black-box solvers. This way solvers will be easily exchangeable.
- Reducing the partitioning error of a coupled problem up to a predefined tolerance by means of quasi-Newton iterations, optionally, accelerated in a multi-fidelity setting using space mapping.
- Applying 2 black-box solvers in a single physical domain. This is a minimal requirement for multi-fidelity acceleration
- Dictating the sequential order of applying the black-box solvers such that the effect of the chosen order on the performance can be easily investigated.

Thus, the pinnacle of this project is the design and development of CASMIR, hence, the main research objective can be formulated as:

> *"To develop a reuseable solver-agnostic coupling framework for the partitioned simulation of strongly-coupled fluid-structure interactions."*

In order to reach this goal the following steps are taken:

- Perform a literature review to identity various quasi-Newton algorithms and become acquainted with those that are most suitable for implementation into CASMIR, alongside studying the Aggressive Space Mapping method.
- Develop a generic quasi-Newton-based coupling framework and implement the elected quasi-Newton algorithms for (inverse-)Jacobian approximation. Extend the framework for multi-fidelity coupling and implement ASM-based variants of the quasi-Newton algorithms.
- Assess the performance of all implemented coupling algorithms (both mono-fidelity and multi-fidelity ones) relative to IQN-ILS(0).

1.4 Overview

This thesis can be divided into three functional parts. The first part (consisting of Chapter 2 through Chapter 4) has a focus on algorithms and the theory supporting the algorithms. Chapter 2 discusses in detail the three fundamental pillars of partitioned simulations (time stepping, sub-iterating and interpolation). Mono-fidelity coupling algorithms used to reduce the interface residual in a partitioned simulation are presented in Chapter 3. Multi-fidelity accelerated coupling algorithms based on the incorporation of algorithms from Chapter 3 into

the Aggressive Space Mapping method are presented in Chapter 4.

The second part consists of Chapter 5, where the design philosophy and code structure of the coupling framework, CASMIR, is discussed in great detail.

The third and final part (consisting of Chapter 6 through Chapter 8) has a focus on the application and the benchmarking of CASMIR. Test cases are defined in Chapter 6. Results obtained from using CASMIR are collected in Chapter 7. Conclusions and recommendations are given in Chapter 8.

Chapter 2

Fundamentals of partitioned fluid-structure interaction simulations

Introduction

Pursueing a partitioned approach to simulate fluid-structure interactions (FSI) leads to requiring at least one solver for the fluid dynamics and one solver for the structure dynamics. These solvers usually employ a time stepping approach resulting in solving (partial) differential equations for the state of the respective systems for multiple time steps. Due to the sequential nature of the partitioned approach, a so-called partitioning error is introduced which is to be reduced by employing an iterative coupling procedure. This, however, implies solving the flow equations and the structure equations multiple times per time step. Therefore, efficient time schemes and coupling algorithms are required to minimise, respectively, the number of time steps and the number of coupling iterations per time step.

An introduction to time integration and efficient high-order time schemes is given in Section 2.1. The coupling procedure required at each time step to reduce the partitioning error is discussed in detail in Section 2.2. Finally, interpolation of interfacial quantities is briefly touched upon in Section 2.3 in the context of non-matching meshes at the fluid-structure interface.

2.1 Time stepping

Time stepping is a popular method used in the Computational Fluid Dynamics (CFD) and the Computational Structural Dynamics (CSD) communities to numerically solve the (partial) differential equations that govern dynamical systems. Time stepping boils down to solving for the state of a dynamical system at a finite number of time levels. Large time steps are crucial in maintaining the computational effort feasible when simulating models of high-fidelity. High-order implicit time schemes are required to ensure both accurate solutions and stability of time stepping when employing large steps.

2.1.1 Time integration

To apply time stepping, first the time domain must be discretised into a finite number of time levels. Starting from an initial condition at time level $n = 0$, the states at successive time levels are computed by means of numerical integration. The specific procedure used to carry out the numerical integration is called a time scheme. In an explicit time scheme, the unknown state at time level $n + 1$ is computed as function of known states at previous time levels. An implicit time scheme also includes an dependency on the unknown state at time level $n + 1$. Take for instance the canonical form of a first-order differential equation

$$\dot{\mathbf{y}} = f(t, \mathbf{y}), \tag{2.1}$$

where $\mathbf{y}$ contains the spatially discretised state of a differential system. Applying a single-step time scheme on (2.1) results in

$$\text{explicit Euler scheme}: \quad \frac{\mathbf{y}^{n+1} - \mathbf{y}^n}{\Delta t} = f(t^n, \mathbf{y}^n), \tag{2.2}$$

$$\text{implicit Euler scheme}: \quad \frac{\mathbf{y}^{n+1} - \mathbf{y}^n}{\Delta t} = f(t^{n+1}, \mathbf{y}^{n+1}), \tag{2.3}$$

where $\Delta t = t^{n+1} - t^n$ is the time step.

Explicit time schemes are conditionally stable at best. Implicit time schemes can also be unconditionally stable, however, this comes at the cost of needing the solution of a system of (non)linear equations as observed from (2.3). When employing large time steps, implicit time schemes are preferred over explicit time schemes due to the stringent stability conditions of the latter, eventhough, implicit schemes are more expensive to apply. Another point to consider is the order of accuracy of a time scheme. The Euler schemes are only first-order accurate, i.e.

$$\mathbf{y}^{n+1} - \mathbf{y}^n = \mathcal{O}(\Delta t). \tag{2.4}$$

Hence, a large time step could lead to very inaccurate solutions. Thus, although, the implicit Euler scheme is unconditionally stable, its application provides inaccurate solutions when using relatively large time steps.

In terms of stability an implicit scheme is preferred. In terms of accuracy a high-order time scheme is preferred. To satisfy both preferences, high-order accurate implicit time schemes are needed. High-order accuracy is also beneficial in terms of efficiency. This follows from a high-order scheme being able to provide the same level of accuracy as a low-order scheme while employing a lower number of time steps, resulting in solving the discretised system of equations less often.

2.1.2 Higher-order schemes

High-order accurate implicit time schemes can be constructed in two ways. The first approach is to compute the state at time level $n+1$ as function of the states at more than one time level.

This is known as linear multi-step. The second approach is to define intermediate stages in between t^n and t^{n+1} and to compute the state at time level $n+1$ as a function of the states at the intermediate stages. Such time schemes are called multi-stage schemes.

Linear multi-step schemes

The general form of a s-step linear mult-step scheme is given by

$$\begin{aligned}\frac{\mathbf{y}^{n+s} + a_{s-1}\mathbf{y}^{n+s-1} + ... + a_0\mathbf{y}^n}{\Delta t} &= b_s f(t^{n+s}, \mathbf{y}^{n+s}) + b_{s-1} f(t^{n+s-1}, \mathbf{y}^{n+s-1}) + \\ &\quad ... + b_0 f(t^n, \mathbf{y}^n)\end{aligned} \tag{2.5}$$

A popular class of linear multi-step schemes for stiff differential equations are the Backward Differencing (BDF) schemes. The general BDF form is obtained from (2.5) by setting $b_{s-1} = ... = b_0 = 0$, i.e.

$$\frac{\mathbf{y}^{n+s} + a_{s-1}\mathbf{y}^{n+s-1} + ... + a_0\mathbf{y}^n}{\Delta t} = b_s f(t^{n+s}, \mathbf{y}^{n+s}). \tag{2.6}$$

Unfortunately, BDF schemes of order higher than 2 are not A-stable.

Definition [16]. *A s-step scheme is called A-stable, if all solutions of (2.5) tend to zero, as $n \to \infty$, when the scheme is applied with a fixed positive Δt to any differential equation of the form*

$$\dot{\mathbf{y}} = c\mathbf{y}, \tag{2.7}$$

where c is a complex constant with negative real part.

Linear multi-step schemes of order higher than 2 should therefore be applied with great care or avoided altogether.

Multi-stage schemes

Runge-Kutta methods are a popular family of multi-stage schemes, where the state at time level $n+1$ is computed as a linear combination of the states at the intermediate stages between timelevels n and $n+1$.

The state at stage k in a s-stage Runge-Kutta method is denoted by $\mathbf{y}^{(k)}$ and it is obtained from

$$\frac{\mathbf{y}^{(k)} - \mathbf{y}^n}{\Delta t} = \sum_{i=1}^{s} a_{ki} f(t^{(i)}, \mathbf{y}^{(i)}), \tag{2.8}$$

where $t^{(i)} = t^n + c_i \Delta t$ and $c_i = \sum_i a_{ki}$. The state at time level $n+1$ is then computed as a linear combination of the states at the s stages

$$\frac{\mathbf{y}^{n+1} - \mathbf{y}^n}{\Delta t} = \sum_{j=1}^{s} b_j f(t^{(j)}, \mathbf{y}^{(j)}), \tag{2.9}$$

where b_k are weights.

A Runge-Kutta method is uniquely defined by specifying values for a_{ij} and b_j. These coefficients are usually compactly collected into a Butcher-Tableau as shown in Table 2.1.

Table 2.1: Butcher Tableau for a general s-stage Runge-Kutta method

c_1	a_{11}	$\dots$	a_{1s}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	$\dots$	a_{ss}
	b_1	$\dots$	b_s

An efficient sub-family of RK schemes used for stiff problems are the ESDIRK schemes. ESDIRK stands for **E**xplicit first stage ($a_{1i} = 0$), **S**ingle diagonal ($a_{kk} = \gamma$), **D**iagonally **I**mplicit ($a_{ki} = 0,\ \forall i > k$), **R**unge-**K**utta. A general s-stage ESDIRK scheme is defined by the Butcher Tableau in Table 2.2.

Table 2.2: Butcher Tableau of a 4-stage ESDIRK scheme

c_1	0	0	0	0
$\vdots$	a_{21}	γ	0	0
$\vdots$	$\vdots$	$\ddots$	$\ddots$	0
c_s	a_{s1}	$\vdots$	a_{ss-1}	γ
	b_1	b_2	b_3	b_4

The benefits of an ESDIRK scheme when compared to other RK schemes are:

- Explicit first stage
  Allows stages to be second-order accurate, otherwise at least the first stage is first-order accurate
- Single diagonal
  The value γ is chosen such that iterative convergence is optimised
- Diagonally implicit
  Stages can be solved sequentially instead of simultaneously

In comparison to BDF schemes, the only downside of ESDIRK schemes (and RK schemes in general) is that the system of differential equations are solved multiple times per time level. This increases the computational work substantially. Unlike BDF schemes however, ESDIRK schemes can attain any order of accuracy while preserving L-stability. The possbility to succesfully select larger time steps must then compensate for the additional amount of work for an ESDIRK scheme to be computationally efficient.

2.2 Coupling iterations

Partitioning a FSI problem results in solving for the flow state separate from solving for the structure state at each time step. In reality, however, the two physical domains influence each other simultaneously in time, whereas the numerical counterpart is reduced to a sequential process. This introduces an additional error term, the so-called partitioning error (also known as the coupling error), to the partitioned numerical simulation. A coupling algorithm is required at each time step to reduce the partitioning error up to a predefined tolerance. The coupling algorithm instigates an iterative coupling process wherein the flow equations and the structure equations are solved multiple times per time step. A low number of coupling iterations is crucial in maintaining partitioned simulations as a feasible option. Therefore, efficient coupling algorithms are needed which converge in a relatively low number of coupling iterations.

The framework used in this thesis to reduce the partitioning error relies heavily on Newton's method for root-finding problems. Therefore, in this section, the theory behind Newton's method is elaborated first; its application to partitioned FSI problems is discussed thereafter.

2.2.1 Newton's method for vector functions

Newton's method (also known as the Newton-Raphson method) is an iterative method for finding roots of non-linear equations. Starting from a first-order Taylor expansion of a non-linear function, the derivation of Newton's method yields a recurrence relation. The recurrence relation is used as an update formula to iteratively find better approximations for the root(s) of the function.

For a graphical representation of Newton's method for scalar equations and a derivation based on Taylor expansions for both scalar and vector equations one may refer to Chapter 13 of [17]. The derivation of Newton's method for vector equations is reproduced here for the sake of completeness.

Exact Newton method

Given $\mathbf{f}$, Newton's method can be used to solve equations of the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \tag{2.10}$$

where $\mathbf{f}$ is a column vector of functions and $\mathbf{x}$ is a column vector of independent variables. Starting from an initial guess $\mathbf{x} = \mathbf{x}^0$, assumed to be close to the exact solution $\mathbf{x} = \mathbf{x}^*$, a first-order Taylor expansion of (2.10) around $\mathbf{x}^0$ yields

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}^0) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^0) \cdot (\mathbf{x} - \mathbf{x}^0). \tag{2.11}$$

From (2.10) it is noted that the left-hand side of (2.11) should equal the zero vector and as such

$$\mathbf{0} = \mathbf{f}(\mathbf{x}^0) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^0) \cdot (\mathbf{x} - \mathbf{x}^0). \tag{2.12}$$

Rewriting (2.12) for $\mathbf{x}$ results in

$$\mathbf{x} = \mathbf{x}^0 + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^0)\right]^{-1} \left(-\mathbf{f}(\mathbf{x}^0)\right). \tag{2.13}$$

The expression denoted by (2.13) is an update formula which can be used to obtain a better approximation of $\mathbf{x}^*$. To further improve the approximation, the $\mathbf{x}$ obtained from (2.13) assumes the role of $\mathbf{x}^0$ in subsequent applications of (2.13) which finally yields the recurrence relation

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} \left(-\mathbf{f}(\mathbf{x}^i)\right), \tag{2.14}$$

used to solve equations of the form given by (2.10). For each iterate $\mathbf{x}^i$ a residual $\mathbf{r}^i$ is defined as

$$\mathbf{r}^i = \mathbf{f}(\mathbf{x}^i). \tag{2.15}$$

The Euclidean norm of $\mathbf{r}^i$ is used as a measure for how good of an approximation $\mathbf{x}^i$ is for $\mathbf{x}^*$. Naturally, if $\|\mathbf{r}^i\|_2 = 0$ then $\mathbf{x}^i \equiv \mathbf{x}^*$. In most cases it is either too computationally expensive or simply impossible due to round-off errors to obtain a residual norm equal to zero. Therefore a tolerance, ϵ, is used such that

$$\mathbf{x}^i := \mathbf{x}^* \quad \text{if} \quad \|\mathbf{r}^i\|_2 \leq \epsilon. \tag{2.16}$$

When the condition in (2.16) is met, the solution is considered to have converged. It is, however, not guaranteed that the solution will converge. Usually, also a cap is set to limit the number of iterations in order to avoid unnecessary computational work in the cases where a stalemate is reached or when the solution converges too slowly.

Quasi-Newton method

Newton's method requires the construction and the evaluation of either the Jacobian of $\mathbf{f}$ or its inverse. There are cases where it is not feasible to construct the (inverse-)Jacobian. Either the (inverse-)Jacobian is too expensive to compute [6, 18] or the function $\mathbf{f}$ is not explicitly known [18, 19]. In those cases the (inverse-)Jacobian must be approximated. All Newton-like algorithms that use an approximated (inverse-)Jacobian are classified as quasi-Newton algorithms.

In general it is more efficient to approximate the inverse-Jacobian rather than the actual Jacobian. The reason follows from (2.14), where a linear system solve is required if one approximates the actual Jacobian, whereas only a single matrix-vector product is needed when the inverse-Jacobian is approximated instead.

2.2.2 Newton's method applied to FSI problems

Coupling algorithms based on Newton's method have been successfully applied to reduce the partitioning error in the partitioned simulation of fluid-structure interactions [18, 19, 8, 9]. To establish the general framework, the coupling problem must be expressed in the form of a root-finding problem similar to (2.10).

Definition of common operators

Several operators have to be defined in order to formulate the root-finding form of a partitioned FSI simulation. The first two are the solver operators $\mathcal{F}$ and $\mathcal{S}$ which correspond, respectively, a black-box flow solver and a black-box structure solver. The solver operators are defined as

$$\boldsymbol{\eta} = \mathcal{F}(\boldsymbol{\xi}) \tag{2.17}$$

and

$$\boldsymbol{\xi} = \mathcal{S}(\boldsymbol{\eta}), \tag{2.18}$$

where $\boldsymbol{\eta}$ is (a part of) the state of the flow and $\boldsymbol{\xi}$ is (a part of) the state of the structure. The exact definitions of $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ in terms of physical quantities depend on the governing equations which the solvers are solving and on the conditions used to enforce the coupling of the partitioned domains.

For example, while $\boldsymbol{\eta}$ could represent an interfacial pressure field or an interfacial velocity field or both if the flow is governed by the Navier-Stokes equations, $\boldsymbol{\eta}$ could also represent an interfacial potential field if the flow is governed by the full-potential equations. The exact definitions of $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ are, however, irrelevant for the current discussion.

The solver operators are applied alternately, to simulate the fluid-structure interaction in a partitioned fashion. The coupling procedure during each time step can start from either an initial guess for the flow state or an initial guess for the structure state. A single sequential evaluation of both solver operators is denoted by the FSI operator, $\mathcal{H}$, defined as

$$\tilde{\mathbf{x}} = \mathcal{H}(\mathbf{x}) = \begin{cases} \mathcal{S} \circ \mathcal{F}(\mathbf{x}), & \mathbf{x} = \boldsymbol{\xi}, \\ \mathcal{F} \circ \mathcal{S}(\mathbf{x}), & \mathbf{x} = \boldsymbol{\eta}, \end{cases} \tag{2.19}$$

where the top definition corresponds the case in which the coupling procedure starts from a flow solve around an initial structure state, whereas the bottom definition corresponds to the case in which the coupling procedure starts from a structure solve around an initial flow state.

After a single evaluation, in general, $\mathbf{x} \neq \mathcal{H}(\mathbf{x})$ will hold due to the presence of a partitioning error. Therefore a residual operator can be defined as

$$\mathbf{r} = \mathcal{R}(\mathbf{x}) \equiv \mathcal{H}(\mathbf{x}) - \mathbf{x} \equiv \tilde{\mathbf{x}} - \mathbf{x}. \tag{2.20}$$

Quasi-Newton update

To cast the coupling problem in the form of a root-finding problem similar to (2.10), the residual operator is simply equated to zero,

$$\mathcal{R}(\mathbf{x}) = \mathbf{0}. \tag{2.21}$$

Using the same derivation as given in Section 2.2.1, applying Newton's method on (2.21) yields

$$\begin{aligned} \mathbf{x}^{i+1} &= \mathbf{x}^i + \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} \left(-\mathcal{R}(\mathbf{x}^i)\right) \\ &\mathbf{x}^i + \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} \left(-\mathbf{r}^i\right). \end{aligned} \tag{2.22}$$

Expanding the Jacobian in (2.22) using the top definition of (2.19) results in

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \left[\left(\frac{\partial \mathcal{S}}{\partial \mathbf{x}} \circ \mathcal{F}(\mathbf{x}^i)\right) \cdot \frac{\partial \mathcal{F}}{\partial \mathbf{x}}(\mathbf{x}^i) - I\right]^{-1} \left(-\mathcal{R}(\mathbf{x}^i)\right). \tag{2.23}$$

According to (2.23), knowledge of the implementation of the flow solver and the structure solver is needed to apply Newton's method. In order to develop a solve-agnostic coupling framework where external solvers can be easily interchanged a quasi-Newton algorithm is used instead. Hence, the external solvers are assumed to be black boxes. This way, the implementation of the coupling algorithm is independent of the external solvers used to simulate the relevant physical domains.

Generic coupling framework

The application of a quasi-Newton algorithm in the reduction of the coupling error of a partitioned FSI simulation revolves around the approximation of the Jacobian,

$$J_i = \frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i), \tag{2.24}$$

or the inverse-Jacobian,

$$J_i^{-1} = \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1}, \tag{2.25}$$

for each coupling iteration of a time step. During a time step, coupling iterations are performed until

$$\| \mathcal{R}(\mathbf{x}^i) \|_2 \equiv \| \mathbf{r}^i \|_2 \leq \epsilon, \tag{2.26}$$

where ϵ is a predefined tolerance, or until the number of coupling iterations exceed a predefined limit. An overview of a generic quasi-Newton coupling procedure at time step n, independent of the order in which the solver operators are evaluated, is summarised in Algorithm 1.

Algorithm 1 Generic quasi-Newton framework

1: Set $i = 0$
2: Define $\mathbf{x}^0$ as an extrapolation of states at previous time levels
3: Evaluate the residual operator: $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
4: If $\| \mathbf{r}^i \|_2 \leq \epsilon$ or $i \equiv i_{\max}$ go to line 9 else continue with next line
5: Compute the Jacobian $\frac{\partial R}{\partial \mathbf{x}}$ or the inverse-Jacobian $\left[\frac{\partial R}{\partial \mathbf{x}}\right]^{-1}$
6: Update the state: $\mathbf{x}^{i+1} = \mathbf{x}^i - \left[\frac{\partial R}{\partial \mathbf{x}}\right]^{-1} \mathbf{r}^i$
7: Set $i = i + 1$
8: Go to line 3
9: Set $\mathbf{x}^* = \mathbf{x}^i$
10: Sync $\mathbf{x}^*$ across solvers and proceed to next time step

Algorithm 1 concludes the elaboration of the application of Newton's method to the partitioned simulation of FSI problems. Algorithms used to perform step 5 of Algorithm 1 are presented in Chapter 3.

2.3 Non-matching meshes

When the characteristic time and length scales of the dynamics in the fluid domain differ greatly from the dynamics in the structure, it is a natural choice to use a different mesh for each domain. Unnecessary computational work can be avoided by employing a coarser mesh for the domain in which the characteristic scales are larger. This however leads to non-matching meshes at the fluid-structure interface (i.e. gaps and/or overlappings and/or different number of nodes at the interface between the two meshes). Interpolation is then needed to transfer state values defined at the interface of one mesh to the interface of the other mesh.

As will be discussed later on in Chapter 5, `Flecs` is a library written in C for MPI-based data communication. The library will be used by the coupling framework developed in this thesis (among others) to perform interpolations. `Flecs` includes the Nearest Neighbour (NN) and the Radial Basis Function (RBF) interpolation methods. Therefore these methods will be discussed next.

Nearest neighbour interpolation

The simplest interpolation method is Nearest Neighbour (NN) interpolation. At a node where a value is unknown, the closest neighbouring node on the interface of the opposing mesh is selected and the value known at that node is copied.

To express this mathematically, let Γ_f denote the fluid-side of the fluid-structure interface and let Γ_s denote the structure-side of the fluid-structure interface. Furthermore, let $\mathbf{x}_{f_i} =$

$[x_{f_i}\ y_{f_i}\ z_{f_i}]^\mathsf{T}$ denote the coordinate of node i on Γ_f and similarly let $\mathbf{x}_{s_j} = [x_{s_j}\ y_{s_j}\ z_{s_j}]^\mathsf{T}$ denote the coordinate of node j on Γ_s. An unknown value at node i on Γ_f is set equal to the known value at the nearest neighbour node j on Γ_s, where node j is defined as

$$j = \underset{1 \leq k \leq N^s}{\arg\min} \{\| \mathbf{x}_{s_k} - \mathbf{x}_{f_i} \|\} \tag{2.27}$$

and N^s is the number of nodes on Γ_s.

NN is quite accurate when interpolating values from a fine grid to a coarse grid, however, the opposite is true for the reverse. This is visualised in Figure 2.1. For each node on the coarse grid there is a nearby node on the fine grid. Due to the large difference in nodes between both grids, multiple nodes on the fine grid receive a value from the same node on the coarse grid. This results in a smooth interpolation when going from a fine grid to a coarse grid and in a staircase-like interpolation when going from a coarse grid to a fine grid.

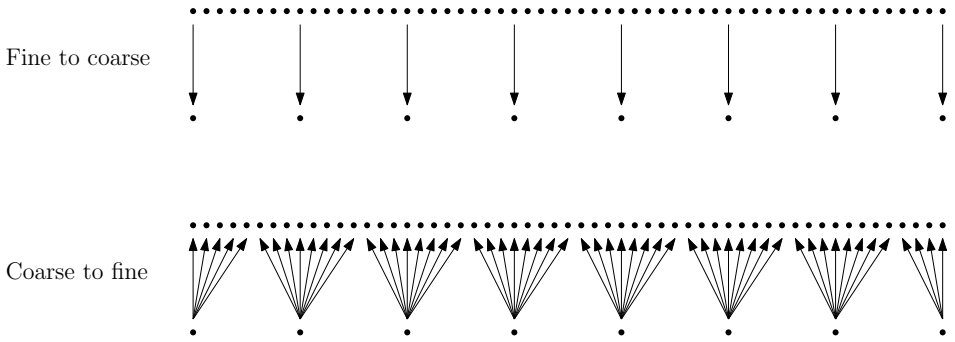


Figure 2.1: Nearest Neighbour interpolation from a fine grid to a coarse grid (top) and from a coarse grid to a fine grid (bottom).

Radial basis function interpolation

A more sophisticated interpolation can be obtained from using radial basis functions (RBF) [10]. RBF interpolation uses a sum of basis functions, ϕ, with respect to the Euclidean distance between a data point and a point of interest. A typical interpolation of displacements at a fluid-structure interface is given by

$$u_i(\mathbf{x}) = \sum_{j=1}^{n_s} \gamma_j \phi(||\mathbf{x} - \mathbf{x}_{s_j}||) + p(\mathbf{x}), \quad i = \{f,\, s\}. \tag{2.28}$$

where x_{s_j} is the coordinate of a data point, in this case a point on the structure side of the interface, p is a polynomial and γ_j are weighting coefficients. The addition of the polynomial is optional, however, it ensures that rigid body translations are recovered exactly *iff* a linear polynomial is used. The weighting coefficients and the polynomial are determined by satisfying

$$u_s(x_{s_j}) = \mathbf{u}_{s_i} \tag{2.29}$$

and

$$\sum_{j=1}^{n_s} \gamma_j q(x_{s_j}) = 0 \quad \forall q \,|\, \partial q \leq \partial p. \tag{2.30}$$

The conditions defined in (2.29) and (2.30) yield an algebraic system that provides the weighting coefficients and the polynomial coefficients upon solving

$$\begin{pmatrix} \mathbf{u}_s \\ \mathbf{0} \end{pmatrix} = \begin{bmatrix} M_{ss} & P_s \\ (P_s)^T & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix}, \tag{2.31}$$

where $\boldsymbol{\beta}$ contains the polynomial coeffcients, $M_{ss} \in \mathbb{R}^{n_s \times n_s}$ contains the evaluations of $\phi(||\mathbf{x}_{s_i} - \mathbf{x}_{s_j}||)$ and $P_s \in \mathbb{R}^{n_s \times 4}$ is a matrix with row j given by $[1 \quad x_{s_j} \quad y_{s_j} \quad z_{s_j}]$.

Defining M_{fs} and P_f in a similar fashion then yields the RBF interpolation for the displacements on the fluid side of the interface

$$\mathbf{u}_f = \begin{bmatrix} M_{fs} & P_f \end{bmatrix} \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix}. \tag{2.32}$$

The computational cost associated with radial basis function interpolation scales rapidly with the number of interface points. Compact support [20] can be used to reduce the computational cost. The compact support consists of a circle in 2D or a sphere in 3D with a predefined radius. The centre of the circle/sphere is placed on top of a point of interest. Then only the data points residing within the circle/sphere are used for the interpolation at the point of interest.

The optimum radius is problem-specific and follows from a trade-off between the accuracy and the computational cost of the interpolation.

For examples and a review of basis functions (with either global or local support) the reader can refer to [10, 20].

Chapter 3

Quasi-Newton algorithms for coupling in time

Introduction

The fundamentals of the partitioned simulation of a FSI problem were discussed in Chapter 2. A generic overview was presented for the quasi-Newton framework used to iteratively reduce the partitioning error. As previously stated, the quasi-Newton framework revolves around the approximation of either the Jacobian or inverse-Jacobian of the residual operator. In this chapter several coupling algorithms are introduced, used to compute the neccessary approximations.

Fixed-point iteration methods like Gauss-Seidel and Aitken Δ^2 underrelaxtion have been used extensively as coupling algorithms with a varying degree of success. These methods are introduced in Section 3.1. More recently the quasi-Newton algorithm known as IQN-ILS has become the state-of-the-art in strongly-coupled FSI simulations. Quasi-Newton algorithms are discussed in Section 3.2. Finally an overview of these algorithms is given in Section 3.3.

3.1 Coupling algorithms of the fixed-point iteration type

Fixed-point iteration methods are used to solve equations of the form

$$\mathbf{x} = \mathbf{f}(\mathbf{x}), \tag{3.1}$$

where the solution $\mathbf{x} = \mathbf{x}^*$ is found by iterating according to

$$\mathbf{x}^{i+1} = \mathbf{f}(\mathbf{x}^i). \tag{3.2}$$

Regular fixed-point iteration methods are quite unstable. To improve the stability a relaxation parameter $\omega \in (0,\, 1)$ is introduced. For stability it is important to choose a small value for ω

[21]. Although, when using a small value only a small percentage of the new information is used for the update which can lead to lower convergence rates [21].

Instead of evaluating $\mathbf{x}^{i+1}$ directly, as regular fixed-point iteration methods like Gauss-Seidel do, relaxation iterations evaluate an approximation

$$\tilde{\mathbf{x}}^{i+1} = \mathbf{f}(\mathbf{x}^i). \tag{3.3}$$

The new iterate $\mathbf{x}^{i+1}$ is then defined as

$$\mathbf{x}^{i+1} = \omega\tilde{\mathbf{x}}^{i+1} + (1-\omega)\mathbf{x}^i. \tag{3.4}$$

The relaxation update can be rewritten as

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \omega(\tilde{\mathbf{x}}^{i+1} - \mathbf{x}^i), \tag{3.5}$$

where $\tilde{\mathbf{x}}^{i+1} - \mathbf{x}^i$ is the residual, $\mathbf{r}^i$, such that

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \omega\mathbf{r}^i. \tag{3.6}$$

In the form given by (3.6), the relaxation update is reminiscent of the update given by Newton's method. Thus, even if fixed-point iteration methods are used to solve equations of the form given by (3.1), they can be recast into a form such that they fit into a quasi-Newton framework. Fixed-point iteration methods that can be written as (3.6), provide Jacobian approximations of the form $-I/\omega$ and inverse-Jacobian approximations of the form $-\omega I$, where I is the identity matrix. This follows from transforming (3.6) into a form equivalent to (2.22):

$$\begin{aligned}
\mathbf{x}^{i+1} &= \mathbf{x}^i + \omega\mathbf{r}^i \\
&= \mathbf{x}^i - \omega(-\mathbf{r}^i) \\
&= \mathbf{x}^i - \omega I(-\mathbf{r}^i) \\
&= \mathbf{x}^i + \underbrace{[-\omega I]}_{\text{inverse-Jacobian}}(-\mathbf{r}^i) \\
&= \mathbf{x}^i + \underbrace{[-I/\omega]}_{\text{Jacobian}}{}^{-1}(-\mathbf{r}^i).
\end{aligned} \tag{3.7}$$

3.1.1 Gauss-Seidel

The Gauss-Seidel method is obtained from (3.6) by setting the relaxation parameter ω equal to one. Hence, the inverse-Jacobian approximation is then simply a diagonal matrix with -1 on the diagonal:

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} = -I. \tag{3.8}$$

As the inverse-Jacobian in (3.8) is independent of the iterate $\mathbf{x}^i$, the Gauss-Seidel approximation performs poorly in strongly-coupled FSI simulations; it either converges slowly or not at all [22]. Gauss-Seidel is usually only suited for weakly-coupled FSI simulations which require only a single coupling iteration per time step.

An implementation of the Gauss-Seidel algorithm consistent with the procedure laid out in Algorithm 1 is presented in Algorithm 2.

Algorithm 2 Gauss-Seidel

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$
$i = 0$
$\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
if $||\mathbf{r}^0|| \leq \epsilon$ **then**
  return $\mathbf{x}^0$
else
  $\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{r}^0$
end if
while $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
  $i = i + 1$
  $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
  $\mathbf{x}^{i+1} = \mathbf{x}^i + \mathbf{r}^i$
end while
return $\mathbf{x}^{i+1}$

3.1.2 Aitken Δ^2 underrelaxation

Choosing a value for ω close to 1, results in an unstable method similar to Gauss-Seidel, choosing a value close to 0 results in a stable yet slowly converging method. Nonetheless, keeping ω fixed is a poor choice [21] for strongly-coupled FSI problems due to the independence of the (inverse-)Jacobian on the iterate $\mathbf{x}^i$. To improve both the stability and the convergence rate simultaneously, an adaptive relaxation parameter is preferred [9, 21].

In the past, the Aitken Δ^2 underrelaxation was the state-of-the-art for strongly-coupled partitioned FSI simulations. It uses an adaptive relaxation parameter which depends on the current and two previous solutions iterates [21]. The relaxation parameter is computed as

$$\omega^i = -\omega^{i-1} \cdot \frac{(\Delta\mathbf{r}^i)^\mathsf{T}\mathbf{r}^{i-1}}{(\Delta\mathbf{r}^i)^\mathsf{T}\Delta\mathbf{r}^i}, \tag{3.9}$$

where $\Delta\mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^{i-1}$ is the difference between the current residual, $\mathbf{r}^i$ and the previous residual, $\mathbf{r}^{i-1}$. The inverse-Jacobian approximation is then given by

$$\left[\frac{\partial\mathcal{R}}{\partial\mathbf{x}}(\mathbf{x}^i)\right]^{-1} = -\omega^i I. \tag{3.10}$$

An implementation of the Aitken Δ^2 underrelaxation coupling algorithm can be obtained from a Gauss-Seidel implementation with minor modifications. As seen in Algorithm 3, the

modifications only incorporate the introduction of a relaxation parameter and the adaptive control thereof.

Algorithm 3 Aitken Δ^2 underrelaxation

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$, ω^0
$i = 0$
$\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
if $||\mathbf{r}^0|| \leq \epsilon$ **then**
  return $\mathbf{x}^0$
else
  $\mathbf{x}^1 = \mathbf{x}^0 + \omega^0 \mathbf{r}^0$
end if
while $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
  $i = i + 1$
  $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
  $\Delta\mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^{i-1}$
  $\omega^i = \omega^{i-1} \cdot \dfrac{(\Delta\mathbf{r}^i)^\mathsf{T}\mathbf{r}^{i-1}}{(\Delta\mathbf{r}^i)^\mathsf{T}\Delta\mathbf{r}^i}$
  $\mathbf{x}^{i+1} = \mathbf{x}^i + \omega^i \mathbf{r}^i$
end while
return $\mathbf{x}^{i+1}$

3.2 Coupling algorithms of the quasi-Newton type

Newton's method is widely used to solve non-linear algebraic equations iteratively. When the (inverse-)Jacobian of the non-linear functions are approximated, the ensuing algorithm falls under the framework of a quasi-Newton algorithm. Approximations are preferred when the computation of the (inverse-)Jacobian is too expensive or strictly neccessary when the non-linear functions are not readily available as is the case when black-box solvers are involved.

Quasi-Newton updates for the state passed as the input argument to the residual operator can be written in the form given by (2.22). As seen from (2.22), it is more efficient to approximate the inverse-Jacobian rather than the Jacobian. This way a linear system solve is avoided in favour of a matrix-vector multiplication.

Compared to fixed-point iterations, faster convergence can be obtained with quasi-Newton algorithms [8, 22]. This is especially the case for the IQN-ILS algorithm which reuses a large residual history to accurately approximate the inverse-Jacobian.

3.2.1 Broyden's method

In applying Newton's method on a scalar equation $f(x) = 0$, the Jacobian reduces to the derivative $f'(x)$. Approximating the derivative with the finite-difference approximation

$$f'(x^i) = \frac{f(x^i) - f(x^{i-1})}{x^i - x^{i-1}}, \tag{3.11}$$

results in the secant method [23]. Broyden's method is an extension of the secant method to non-linear systems [24]. Starting from an initial estimate for the Jacobian, Broyden's method iteratively improves the Jacobian with a rank-one update (i.e. modifying the Jacobian by adding a matrix whose rank is equal to 1)

$$\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i) = \frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1}) + \frac{\Delta \mathbf{r}^i - \frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\Delta \mathbf{x}^i}{(\Delta \mathbf{x}^i)^\mathsf{T} \Delta \mathbf{x}^i}(\Delta \mathbf{x}^i)^\mathsf{T}, \tag{3.12}$$

where $\Delta \mathbf{x}^i = \mathbf{x}^i - \mathbf{x}^{i-1}$ and $\Delta \mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^{i-1}$. To avoid a linear system solve, Broyden suggested in [6] to use Householder's transformation (i.e. a linear transformation describing a reflection in a plane containing the origin) to modify (3.12) such that the inverse-Jacobian is updated instead. The Householder transformation of the rank-one update results in the Sherman-Morrison transformation (for details see [24, 25]) and when applied to Broyden's methods it yields an update for the inverse-Jacobian

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} = \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1} + \frac{\Delta \mathbf{x}^i - \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1} \Delta \mathbf{r}^i}{(\Delta \mathbf{x}^i)^\mathsf{T} \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1} \Delta \mathbf{r}^i} (\Delta \mathbf{x}^i)^\mathsf{T} \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1}. \tag{3.13}$$

Declaring $J_i = \frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)$ and $J_i^{-1} = \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1}$ for the sake of brevity, an implementation of the the Broyden algorithm using the Sherman-Morrison transformation is laid out in Algorithm 4.

Algorithm 4 Broyden

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$
1: $i = 0$
2: $\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
3: **if** $||\mathbf{r}^0|| \leq \epsilon$ **then**
4: **return** $\mathbf{x}^0$
5: **else**
6: $J_0^{-1} = -I$
7: $\mathbf{x}^1 = \mathbf{x}^0 - J_0^{-1}\mathbf{r}^0$
8: **end if**
9: **while** $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
10: $i = i + 1$
11: $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
12: $\Delta \mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^{i-1}$
13: $\Delta \mathbf{x}^i = \mathbf{x}^i - \mathbf{x}^{i-1}$
14: $J_i^{-1} = J_{i-1}^{-1} + \frac{\Delta \mathbf{x}^i - J_{i-1}^{-1}\Delta \mathbf{r}^i}{(\Delta \mathbf{x}^i)^\mathsf{T} J_{i-1}^{-1} \Delta \mathbf{r}^i}(\Delta \mathbf{x}^i)^\mathsf{T} J_{i-1}^{-1}$
15: $\mathbf{x}^{i+1} = \mathbf{x}^i - J_i^{-1}\mathbf{r}^i$
16: **end while**
17: **return** $\mathbf{x}^{i+1}$

3.2.2 IQN-ILS

The Aitken and Broyden algorithms use data from the current and up to two previous iterations to approximate the inverse-Jacobian. Gauss-Seidel on the other hand, uses a constant approximation throughout the entire simulation. Although Aitken and Broyden converge faster than Gauss-Seidel for strongly-coupled interactions due to reusing data from previous iterations, the reuse is quite limited. To obtain even better convergence rates it is crucial to reuse data from even more previous iterations. Based on that notion a quasi-Newton method was developed that uses a least-squares formulation to approximate the inverse-Jacobian . The so-called "interface quasi-Newton algorithm with inverse-Jacobian from least-squares" (IQN-ILS) is the current state-of-the-art for mono-fidelity strongly-coupled partitioned FSI simulations (for benchmarks see [8, 9]). Compared to the previously introduced coupling algorithms, IQN-ILS is somewhat involved. An exposition of the theory supporting IQN-ILS is therefore presented here to clarify the major steps taken by the algorithm.

IQN-ILS

Starting each time step from an initial guess for the structure state (extrapolated from known states at previous time levels) and using $\mathbf{x}$ instead of $\mathbf{d}$ to denote the displacement, the residual in coupling iteration i is defined as

$$\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i) \equiv \mathcal{H}(\mathbf{x}^i) - \mathbf{x}^i = \mathcal{S} \circ \mathcal{F}(\mathbf{x}^i) - \mathbf{x}^i = \tilde{\mathbf{x}}^{i+1} - \mathbf{x}^i, \tag{3.14}$$

where $\mathbf{x}^i$ is the structural displacement at the beginning of coupling iteration i and $\tilde{\mathbf{x}}^{i+1}$ is an approximation for $\mathbf{x}^{i+1}$, the displacement at the beginning of the next iteration. In the course of performing coupling iterations a set of residual vectors,

$$\{\mathbf{r}^i, \mathbf{r}^{i-1}, ..., \mathbf{r}^0\} \tag{3.15a}$$

and a set of approximation vectors,

$$\{\tilde{\mathbf{x}}^{i+1}, \tilde{\mathbf{x}}^i, ..., \tilde{\mathbf{x}}^1\}, \tag{3.15b}$$

are generated. The difference between the oldest and the newest vectors of these sets are denoted by

$$\Delta\mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^0 \tag{3.16a}$$

and

$$\Delta\tilde{\mathbf{x}}^{i+1} = \tilde{\mathbf{x}}^{i+1} - \tilde{\mathbf{x}}^1 \tag{3.16b}$$

and they form the columns of the matrices

$$V^i = \left[\Delta\mathbf{r}^i, \Delta\mathbf{r}^{i-1}, ..., \Delta\mathbf{r}^0\right] \tag{3.17}$$

and

$$W^i = \left[\Delta\tilde{\mathbf{x}}^{i+1}, \Delta\tilde{\mathbf{x}}^i, ..., \Delta\tilde{\mathbf{x}}^1\right]. \tag{3.18}$$

Ideally, the goal is to converge to $||\mathbf{r}^{i+1}||_2 = 0 \rightarrow \mathbf{r}^{i+1} = \mathbf{0}$, otherwise formulated as $\Delta\mathbf{r} = \mathbf{0} - \mathbf{r}^i$. The vector $\Delta\mathbf{r}$ is approximated as a linear combination of the columns of matrix V^i such that

$$\Delta\mathbf{r} \approx V^i\mathbf{c}^i \tag{3.19}$$

where $\Delta\mathbf{r}$, $\mathbf{c}^i \in \mathbb{R}^u$, $V^i \in \mathbb{R}^{u\times v}$ and $u \geq v$. Due to the constraint $u \geq v$, (3.19) is an overdetermined system of linear equations. If the number of elements in the vector-argument of the residual operator is low or if the number of coupling iterations needed for convergence is high, v might tend to become larger than u. In those cases, to prevent (3.19) from becoming underdetermined, the right-most columns of V^i and W^i are discarded.

A least-squares solution of the overdetermined system of linear equations (3.19) is obtained from

$$\mathbf{c}^i = \left((V^i)^\mathsf{T}V^i\right)^{-1}(V^i)^\mathsf{T}\Delta\mathbf{r}. \tag{3.20}$$

Unfortunately, for a large number of columns in V^i the usage of (3.20) leads to an unstable implementation of the coupling algorithm [9]. Alternatively, a so-called economy-size QR-decomposition of V^i is used instead for the least-squares solution of (3.19). Defining

$$V^i = Q^iR^i \tag{3.21}$$

where $Q^i \in \mathbb{R}^{u\times v}$ is a orthogonal matrix and $R^i \in \mathbb{R}^{v\times v}$ is an upper-triangular matrix, the coefficient vector $\mathbf{c}^i$ is obtained from solving the triangular system

$$R^i\mathbf{c}^i = (Q^i)^\mathsf{T}\Delta\mathbf{r}, \tag{3.22}$$

using back substitution. If matrix V^i is degenerate (i.e. $rank(V^i) < v$) back substitution will fail due to R^i having zeroes on its main diagonal as a consequence. To bypass the degenerate and the almost-degenerate cases a threshold ϵ_{QR} is defined, used to detect (almost-)zero entries in the main diagonal of R^i after which one sets

$$c^i_j = 0 \quad \text{if} \quad |R^i_{jj}| \leq \epsilon_{QR}. \tag{3.23}$$

The vector $\Delta\tilde{\mathbf{x}}$ corresponding to $\Delta\mathbf{r}$ is computed as a linear combination of the columns W^i using the same coefficient vector $\mathbf{c}^i$,

$$\Delta\tilde{\mathbf{x}} = W^i\mathbf{c}^i. \tag{3.24}$$

As a consequence of (3.14),

$$\Delta \mathbf{r} = \Delta \tilde{\mathbf{x}} - \Delta \mathbf{x}. \tag{3.25}$$

Substituting (3.24) into (3.25) and noting $\Delta \mathbf{r} = \mathbf{0} - \mathbf{r} = -\mathbf{r}$, results in

$$\Delta \mathbf{x} = W^i \mathbf{c}^i - \Delta \mathbf{r} = W^i \mathbf{c}^i + \mathbf{r}. \tag{3.26}$$

Hence, the IQN-ILS update finally yields to

$$\mathbf{x}^{i+1} = \mathbf{x}^i + W^i \mathbf{c}^i + \mathbf{r}. \tag{3.27}$$

As observed from (3.27), when using IQN-ILS as a coupling algorithm there is no need to explicitly compute and store the (inverse-)Jacobian. Such a matrix-free implementation as presented in Algorithm 5 is more efficient in terms of both computational work and memory usage, compared to an implementation using an inverse-Jacobian explicitly defined as

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} = W^i (R^i)^{-1} (Q^i)^{\mathsf{T}} - I. \tag{3.28}$$

Algorithm 5 IQN-ILS

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$
$i = 0$
$\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
if $||\mathbf{r}^0|| \leq \epsilon$ **then**
  return $\mathbf{x}^0$
else
  $\tilde{\mathbf{x}}^1 = \mathbf{r}^0 + \mathbf{x}^0$
  $\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{r}^0$
end if
while $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
  $i = i + 1$
  $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
  $\tilde{\mathbf{x}}^{i+1} = \mathbf{r}^i + \mathbf{x}^i$
  $\Delta\mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^0$
  $\Delta\tilde{\mathbf{x}}^{i+1} = \tilde{\mathbf{x}}^{i+1} - \tilde{\mathbf{x}}^1$
  if $i == 1$ **then**
    $V = \left[\Delta\mathbf{r}^i\right]$
    $W = \left[\Delta\tilde{\mathbf{x}}^{i+1}\right]$
  else
    $V = \left[\Delta\mathbf{r}^i \quad V\right]$
    $W = \left[\Delta\tilde{\mathbf{x}}^{i+1} \quad W\right]$
  end if
  $Q,\, R = \mathsf{qr}(V)$
  solve $R\mathbf{c} = -Q^{\mathsf{T}}\mathbf{r}^i$
  $\mathbf{x}^{i+1} = \mathbf{x}^i + W\mathbf{c} + \mathbf{r}^i$
end while
return $\mathbf{x}^{i+1}$

IQN-ILS(r)

While IQN-ILS obtains its relatively high convergence rate from reusing data from previous coupling iterations of the current time step, the convergence rate can be increased even further by reusing data from previous time steps. This leads to IQN-ILS(r), where r indicates that data from r previous time steps is used to compute the update. To obtain IQN-ILS(r) from IQN-ILS, one simply needs to tag the matrices V^i and W^i with the index of the time step in which they were formed. Finally, instead of (3.17) and (3.17) one uses

$$V^i = \left[\Delta\mathbf{r}^i,\, \Delta\mathbf{r}^{i-1},\, ...,\, \Delta\mathbf{r}^0,\, {}^{n-1}V,\, {}^{n-2}V,\, ...,\, {}^{n-r}V\right] \tag{3.29}$$

and

$$W^i = \left[\Delta\tilde{\mathbf{x}}^{i+1},\, \Delta\tilde{\mathbf{x}}^i,\, ...,\, \Delta\tilde{\mathbf{x}}^1,\, {}^{n-1}W,\, {}^{n-2}W,\, ...,\, {}^{n-r}W\right], \tag{3.30}$$

respectively, where one defines

$$ {}^{n}V = \left[\Delta\mathbf{r}^{i},\, \Delta\mathbf{r}^{i-1},\, ...,\, \Delta\mathbf{r}^{0}\right] \tag{3.31} $$

and

$$ {}^{n}W = \left[\Delta\tilde{\mathbf{x}}^{i+1},\, \Delta\tilde{\mathbf{x}}^{i},\, ...,\, \Delta\tilde{\mathbf{x}}^{1}\right]. \tag{3.32} $$

upon convergence at time step n. Note however, that the dimensions of matrices V^i and W^i still have to satisfy $u \geq v$. Accordingly, the right-most columns of V^i and W^i are to be discarded to maintain the constraint.

In the framework of IQN-ILS(r), IQN-ILS can be seen as IQN-ILS(0). The optimal value for r is problem-dependent and cannot be a priori known. However, large values for r can have adverse effects on the convergence rate as data from too many time steps back might not be capable of representing the solution of the current time level. See Algorithm 6 for an overview of the IQN-ILS(r) algorithm.

Algorithm 6 IQN-ILS(r)

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$, V^{prev}, W^{prev}
1: $i = 0$
2: $\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
3: **if** $||\mathbf{r}^0|| \leq \epsilon$ **then**
4: **return** $\mathbf{x}^0$
5: **else**
6: $V = [V^{\mathsf{prev}}]$
7: $W = [W^{\mathsf{prev}}]$
8: $\tilde{\mathbf{x}}^1 = \mathbf{r}^0 + \mathbf{x}^0$
9: $\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{r}^0$
10: **end if**
11: **while** $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
12: $i = i + 1$
13: $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
14: $\tilde{\mathbf{x}}^{i+1} = \mathbf{r}^i + \mathbf{x}^i$
15: $\Delta\mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^0$
16: $\Delta\tilde{\mathbf{x}}^{i+1} = \tilde{\mathbf{x}}^{i+1} - \tilde{\mathbf{x}}^1$
17: $V = \left[\Delta\mathbf{r}^i \quad V\right]$
18: $W = \left[\Delta\tilde{\mathbf{x}}^{i+1} \quad W\right]$
19: $Q,\, R = \mathsf{qr}(V)$
20: solve $R\mathbf{c} = -Q^{\mathsf{T}}\mathbf{r}^i$
21: $\mathbf{x}^{i+1} = \mathbf{x}^i + W\mathbf{c} + \mathbf{r}^i$
22: **end while**
23: **return** $\mathbf{x}^{i+1}$

3.3 Overview of quasi-Newton updates

In this chapter a total of four temporal coupling algorithms were introduced (Gauss-Seidel, Aitken Δ^2 underrelaxation, Broyden and IQN-ILS) for iteratively reducing the partitioning error at a time step. The Broyden and IQN-ILS algorithms are pure quasi-Newton, meaning to say that they were primarily developed to solve root-finding problems via Newton's method while using an approximating for either the Jacobian or the inverse-Jacobian. On the other hand, the Gauss-Seidel and the Aitken Δ^2 underrelaxation algorithms are inherently used to solve fixed-point problems. Nonetheless, fixed-point iteration methods of the form given by (3.6), can be reformulated as a quasi-Newton method.

The Gauss-Seidel algorithm uses a constant inverse-Jacobian throughout the entire simulation, even though the interface residual is a function of the in time varying state of the multiphysical system. Therefore, the Gauss-Seidel algorithm is a poor choice for strongly-coupled partitioned FSI simulations.

Aitken Δ^2 underrelaxation improves upon Gauss-Seidel by using a relaxation parameter which depends on the residuals from the previous two coupling iterations. This way, the inverse-Jacobian depends on the state of the system and as such, Aitken is a more suitable coupling algorithm for strongly coupled problems.

Starting from an initial guess for the Jacobian, Broyden uses a rank-one update to improve the Jacobian iteratively. Through applying the Sherman-Morrison formula, the Broyden algorithm is reformulated to update the inverse-Jacobian instead. Similar to Aitken, Broyden's inverse-Jacobian approximation depends on the state of the system. However, in contrast to Aitken, Broyden provides as approximation a dense matrix, whereas Aitken provides a diagonal matrix. Hence, in terms of FLOPS per iteration and memory usage per iteration, Aitken is somewhat cheaper than Broyden if the properties of the approximation matrices are exploited when implementing the coupling algorithm.

In cases of the aforementioned algorithms, at the most a very short residual history is used to approximate the inverse-Jacobian. IQN-ILS uses the complete residual history of a single time step to approximate the inverse-Jacobian. IQN-ILS(r), even reuses residual histories from r previous time steps. The vast amount of data allows for an increasingly more accurate approximation. The reuse of a large residual history has as direct consequence a severely larger amount computational work and memory usage per coupling iteration relative to other coupling algorithms. Yet, if the computational time needed for computing the approximation is negligible compared to the computational time needed to evaluate the solver operators, the reduction in required coupling iterations overcompensates the additional computational work introduced by reusing a larger residual history.

Finally, this chapter is concluded with an overview of the various approximations for the inverse-Jacobian:

Gauss-Seidel

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} = -I$$

Aitken Δ^2

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} = -\omega^i \cdot \frac{(\mathbf{r}^{i-1})^\mathsf{T} \cdot (\mathbf{r}^i - \mathbf{r}^{i-1})}{(\mathbf{r}^i - \mathbf{r}^{i-1})^\mathsf{T}(\mathbf{r}^i - \mathbf{r}^{i-1})} I$$

Broyen (3.33)

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} = \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1} + \frac{\Delta \mathbf{x}^i - \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1} \Delta \mathbf{r}^i}{(\Delta \mathbf{x}^i)^\mathsf{T} \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1} \Delta \mathbf{r}^i} (\Delta \mathbf{x}^i)^\mathsf{T} \left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^{i-1})\right]^{-1}$$

IQN-ILS

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} = W^i (R^i)^{-1} (Q^i)^\mathsf{T} - I$$

Chapter 4

Multi-fidelity acceleration through Aggressive Space Mapping

Introduction

In Chapter 3, a quasi-Newton framework was introduced for iteratively reducing the temporal coupling error stemming from the sequential fashion used to solve for the states of the flow and structure domains. Coupling algorithms introduced in Chapter 3 are of the mono-fidelity type, i.e., the coupling algorithms use only a single solver per physical domain. The IQN-ILS and especially the IQN-ILS(r) are extremely efficient coupling algorithms usually requiring significantly less coupling iterations to converge compared to Aitken Δ^2 underrelaxation, the previous state-of-the-art for strongly-coupled FSI problems. Nonetheless, when high-fidelity solvers are applied in the partitioned simulation of a strongly-coupled FSI problem, the requirement of multiple solver evaluations per time step forms a huge penalty in terms of computational cost.

Partitioned FSI simulations of strongly-coupled problems using only high-fidelity solvers are not feasible for industrial purposes, where designing is a cyclic process consisting of prototyping, testing, analysing and refining. One way to make such simulations more appealing, computational-wise, is to employ a multi-fidelity coupling algorithm. A multi-fidelity coupling algorithm uses multiple solvers per physical domain of distinct fidelity. Data generated by iterating with the low-fidelity solvers is exploited to reduce the number of coupling iterations needed for the convergence of the high-fidelity solvers. A computational acceleration is then based on the premise that the increase in computational work related to the evaluations of the low-fidelity solvers is negligible compared to the decrease in computational work related to the reduction in the number of evaluations of the high-fidelity solvers.

Space-mapping, a method originally designed for multi-fidelity optimisation, is introduced in this chapter as an iterative procedure accelerator using multiple models of distinct fidelity for the same physical problem. In particular, the space-mapping method is employed here as a reformulated Newton's method, yielding the Aggressive Space-Mapping (ASM) algorithm. Originally, ASM used Broyden's method to approximate Jacobians. In this thesis,

several ASM-variants are developed using the inverse-Jacobian approximations previously introduced in Chapter 3.

The fundamentals of space-mapping are presented in Section 4.1. Subsequently, the application of ASM to FSI problems is elaborated in Section 4.2. This chapter concludes with the establishment of several ASM-variants in Section 4.3 using inverse-Jacobian approximations from Chapter 3.

4.1 Fundamentals of space mapping

Space mapping is essentially the mathematical concept of applying a transformation on a member of a vector space such that the result is a member of another vector space. More specifically, in an engineering context, space mapping is a method used to map between the solution spaces of distinct models of distinct fidelity describing the same physical problem. Space mapping exploits solution data obtained from a computationally cheap, yet inaccurate, low-fidelity model to accelerate the iterative convergence process of a computationally expensive, yet accurate, high-fidelity model.

4.1.1 Definition of the mapping and inverse mapping operators

To clarify the fundamentals of space mapping consider a generic dynamical system. The evolution of the state of the system in time can be simulated accurately using a high-fidelity model or inaccurately using a low-fidelity model. The (vector) space of all possible states obtainable with the high-fidelity model is denoted by X. A high-fidelity model operator, $\mathcal{H}$, is defined as

$$\tilde{\mathbf{x}}^{i+1} = \mathcal{H}(\mathbf{x}^i), \tag{4.1}$$

where $\mathbf{x}^i \in X$ is an estimate for the state of the system governed by the high-fidelity model and $\tilde{\mathbf{x}}^{i+1}$ is an approximation for the next estimate, $\mathbf{x}^{i+1} \in X$. Analogously, the (vector) space of all possible states obtainable with the low-fidelity model is denoted by Z. A low-fidelity model operator, $\tilde{\mathcal{H}}$, is defined as

$$\tilde{\mathbf{z}}^{i+1} = \tilde{\mathcal{H}}(\mathbf{z}^i), \tag{4.2}$$

where $\mathbf{z}^i \in Z$ and $\tilde{\mathbf{z}}^{i+1}$ are defined analogously to their high-fidelity counterparts. Introducing the space mapping operator,

$$\mathcal{P} : X \to Z, \tag{4.3}$$

used to map from a high-fidelity state to a corresponding low-fidelity state; the operator is abstractly defined as

$$\mathbf{z} = \mathcal{P}(\mathbf{x}) \equiv \arg\min_{\mathbf{z} \in Z} \| \tilde{\mathcal{R}}(\mathbf{z}) - \mathcal{R}(\mathbf{x}) \|, \tag{4.4}$$

where

$$\mathcal{R}(\mathbf{x}^i) \equiv \mathcal{H}(\mathbf{x}^i) - \mathbf{x}^i \tag{4.5a}$$

is the residual operator for the high-fidelity model and

$$\tilde{\mathcal{R}}(\mathbf{z}^i) \equiv \tilde{\mathcal{H}}(\mathbf{z}^i) - \mathbf{z}^i \tag{4.5b}$$

is the residual operator for the low-fidelity model.

The mapping operator, $\mathcal{P}$, is said to be perfect *iff* the mapping operator maps the converged state of the high-fidelity model, $\mathbf{x}^*$, to the converged state of the low-fidelity model, $\mathbf{z}^*$. Hence, $\mathcal{P}$ is perfect *iff*

$$\mathcal{P}(\mathbf{x}^*) = \mathbf{z}^*. \tag{4.6}$$

Analogously to (4.4), the inverse of the mapping operator is abstractly defined as

$$\mathbf{x} = \mathcal{P}^{-1}(\mathbf{z}) \equiv \underset{\mathbf{x}\in X}{\arg\min} \left\| \mathcal{R}(\mathbf{x}) - \tilde{\mathcal{R}}(\mathbf{z}) \right\| . \tag{4.7}$$

In terms of computational work, evaluating the inverse mapping operator, $\mathcal{P}^{-1}$, is equivalent to solving the high-fidelity model directly (e.g. solving the high-fidelity model without exploiting any other model). In order to achieve a computational acceleration relative to the direct method, an approximation, $\mathcal{P}_i^{-1}$, for the inverse mapping operator is used instead such that

$$\mathcal{P}^{-1}(\mathbf{z}^*) = \mathbf{x}^* \approx \mathbf{x}^{i+1} = \mathcal{P}_i^{-1}(\mathbf{z}^*). \tag{4.8}$$

The following assumption is made:

$$\lim_{i\to\infty} \{\mathcal{P}_i^{-1}(\mathbf{z}^*)\} = \mathcal{P}^{-1}(\mathbf{z}^*) \Rightarrow \lim_{i\to\infty} \mathbf{x}^i = \mathbf{x}^* \tag{4.9}$$

There are cases where (4.9) does not hold, yet those cases are outside of the scope of this thesis. Interested readers may refer to [2] for an examination of such more complex cases.

The application of space mapping to the partitioned simulation of strongly-coupled FSI problems is the subject of Section 4.2.

4.1.2 Low-fidelity models

The application of space mapping to the simulation of a physical problem yields a multi-fidelity simulation framework. Hence, multiple models of distinct fidelity and capable of describing the same physical problem are needed. Given a high-fidelity model, space mapping does not incorporate any method to select or construct a low-fidelity model. Furthermore, if the applied models are too dissimilar convergence might be a problem; yet if the models are too similar in terms of computational complexity a computational acceleration might not be

achieved.

To date no universally applicable method exist that leads to an "optimal" low-fidelity model. The pairing of an "optimal" low-fidelity model to a given high-fidelity model is therefore a problem-specific task.

Without guaranteeing suitablility, several methods exist for constructing low-fidelity models. Readers may refer to [26] for an extensive listing and additionally consult references therein for more details. In this thesis, however, low-fidelity models are derived from a high-fidelity model by specifically reducing the fidelity of the simulated physics. Several options for this type of low-fidelity models are presented hereafter without delving into their suitability.

Simplify governing equations

Perhaps the most intuitive method, from a physical viewpoint, is to neglect some physical phenomena modeled by the high-fidelity model through the removal of terms in the equations describing the physics or by linearising the equations.

Advantages:

- A single mesh can be used. In some cases it might, however, be more beneficial to use multiple meshes. An example is when viscosity is neglected; a high number of cells near solid boundaries used to capture boundary layers is then no longer needed.

Disadvantages:

- Requires multiple sets of governing equations. If the black-box solvers do not contain multiple models you are forced to either employ multiple software packages or switch completely to a new package.
- Crucial features predicted by the high-fidelity model might no longer be captured.

Raise numerical tolerance

Numerical simulations of physical problems usually involve several predefined tolerances to limit the number of iterations in an iterative process. Raising the value of tolerances is the least user-intensive approach for constructing a low-fidelity model.

Advantages:

- Requires only a single set of governing equations because the employed model(s) remains unchanged in this case.
- Both models are governed by the same differential equations, hence, the same mesh can be used for both models.

Disadvantages:

- The coarse model solver can produce unphysical solutions if the tolerance is met before the solutions have fully converged. This might lead to slower convergence rates or even divergence.

- Governing equations, numerical algorithms and meshes used for both models are the same. The low-fidelity model might not be cheap compared to the high-fidelity model.

Coarse discretisation

Simulations of problems governed by (partial) differential equations usually require the discretisation of the equations, numerical domain and solid structures contained in the numerical domain. A low-fidelity model can be obtained by using coarser discretisations.

Advantages:

- A single set of governing equations can be used. However, discretisations that are too coarse dissallow the solver to capture some physical phenomena predicted by the model. It might be beneficial to use a less complex set of equations for the low-fidelity model to avoid a waste of computational power.
- Requires less storage for low-fidelity mesh and solution data.

Disadvantages:

- Requires multiple meshes, hence, more time is spend on mesh generation and more storage is needed to store the additional mesh.
- Features predicted by the high-fidelity model might not be captured by the low-fidelity model.

4.2 Aggressive Space-Mapping applied to FSI problems

To achieve a computational acceleration relative to the coupling algorithms introduced in Chapter 3, some considerations related to the selection of the flow models and structure models will be introduced shortly. Thereafter, the space-mapping enhanced coupling framework of Aggressive Space Mapping is derived and an additional "auxillary" FSI problem is introduced, required to drive the quasi-Newton updates.

4.2.1 Model selection

The partitioned simulation of fluid-structure interactions requires at least one model for the flow state and one model for the structural deformation. By applying space mapping to the temporal coupling process, the number of required models increases. Restricting the possibilities to a maximum of two models of distinct fidelity per physical domain, applying space mapping to partitioned FSI simulations leads to the three configurations listed in Table 4.1. A tilde is used to denote the model operator of lower fidelity. The low-fidelity models can be derived from the high-fidelity models as discussed in Section 4.1.2.

Table 4.1: Solver configurations for multi-fidelity accelerated partitioned FSI simulations

Configuration	Flow solver operator(s)	Structure solver operator(s)
1	$\mathcal{F}$ and $\tilde{\mathcal{F}}$	$\mathcal{S}$ and $\tilde{\mathcal{S}}$
2	$\mathcal{F}$	$\mathcal{S}$ and $\tilde{\mathcal{S}}$
3	$\mathcal{F}$ and $\tilde{\mathcal{F}}$	$\mathcal{S}$

Based on the configurations listed in Table 4.1, the high- and low-fidelity models for partitioned FSI are listed in Table 4.2.

Table 4.2: Summary of fidelity levels for modeling a multi-physical interaction based on configurations listed in Table 4.1

	Configuration 1	Configuration 2	Configuration 3
High	Couple $\mathcal{F}$ with $\mathcal{S}$	Couple $\mathcal{F}$ with $\mathcal{S}$	Couple $\mathcal{F}$ with $\mathcal{S}$
Low	Couple $\tilde{\mathcal{F}}$ with $\tilde{\mathcal{S}}$	Couple $\mathcal{F}$ with $\tilde{\mathcal{S}}$	Couple $\tilde{\mathcal{F}}$ with $\mathcal{S}$

At the Aerodynamics department, FSI research is usually biased towards the flow-side of the simulation. This implies using an expensive model for the flow physics and a considerably less expensive model for the structure physics. Hence, in this thesis, mainly configuration 3 is considered. Therefore it is crucial to be aware of the properties of well-known flow models in terms of physical accuracy and computational complexity as observed in Figure 4.1.

As high-fidelity model for the flow, the Euler equations can be selected when viscosity and heat transfer can be neglected, otherwise the Navier-Stokes equations should be used. If small-scale effects, both time-wise and spatial-wise, play a non-negligible role one should opt for Favre/Reynolds-Averaged Navier-Stokes equations (FANS/RANS). Only if the flow field is very complex (flow separation, shock/boundary layer interaction etc.) and a high resolution is required, should Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) be considered.

Currently, the application of LES and DNS is quite prohibitive for large-scale industrial purposes. Full and linearised potential flow equations reside, however, on the other side of the spectrum and might not provide sufficient accuracy and should therefore not be applied as high-fidelity model for industrial purposes outside of a conceptual design phase.

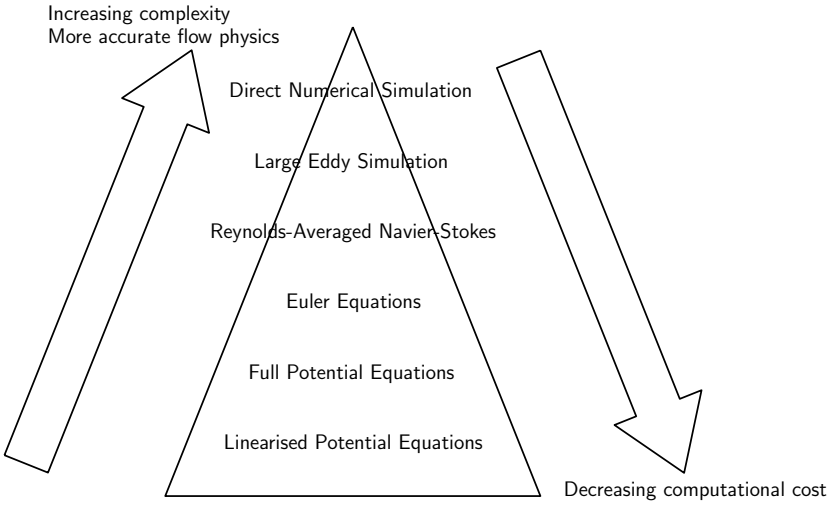


Figure 4.1: Hierarchy of flow models

4.2.2 Aggressive Space-Mapping algorithm

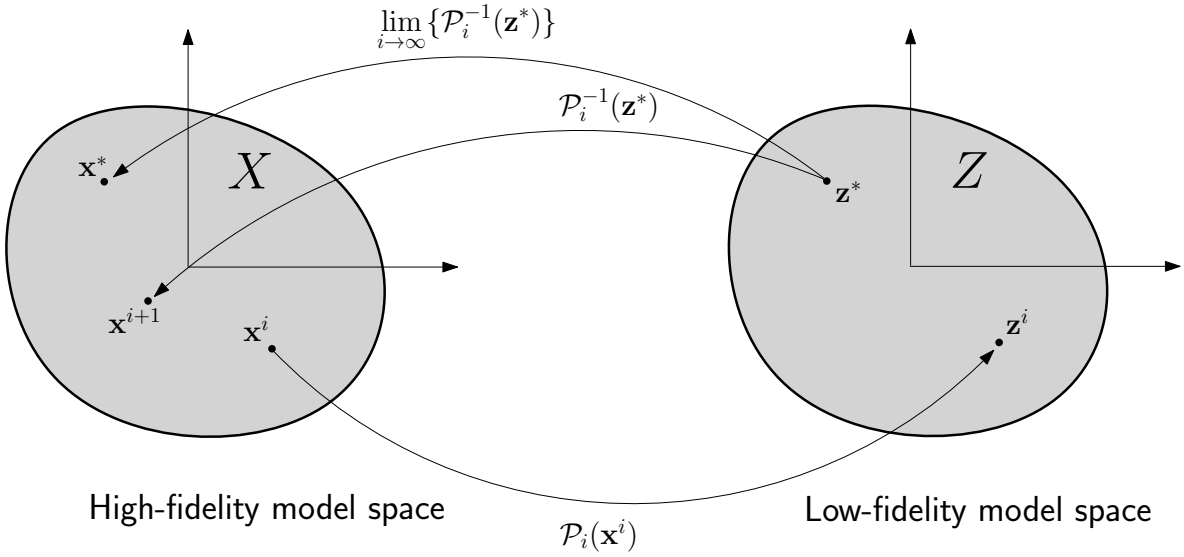


Figure 4.2: Mapping between solution spaces of models of distinct fidelity

In general, both the mapping and the inverse mapping operators are not explicitly known. ASM therefore uses approximations derived form the first-order Taylor expansion of the mapping operator, $\mathcal{P}$. The approximations are improved iteratively and upon convergence, $\mathbf{z}^*$ is mapped to $\mathbf{x}^*$ as depicted in Figure 4.2.

The first-order Taylor expansion of the mapping operator results in the approximation

$$\mathcal{P}_i(\mathbf{x}) \approx \mathcal{P}(\mathbf{x}^i) + \frac{\partial \mathcal{P}}{\partial \mathbf{x}}(\mathbf{x}^i) \cdot (\mathbf{x} - \mathbf{x}^i). \tag{4.10}$$

Noting that $\mathcal{P}_i(\mathbf{x}) \approx \mathbf{z}$, rewriting for $\mathbf{x}$ results in

$$\begin{aligned} \mathbf{x} &\approx \mathbf{x}^i + \left[\frac{\partial \mathcal{P}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} \left(\mathcal{P}_i(\mathbf{x}) - \mathcal{P}(\mathbf{x}^i)\right) \\ &\approx \mathbf{x}^i + \left[\frac{\partial \mathcal{P}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} \left(\mathbf{z} - \mathbf{z}^i\right). \end{aligned} \tag{4.11}$$

Also noting that $\mathbf{x} \equiv \mathcal{P}^{-1}(\mathbf{z})$ and $\mathbf{x}^i \equiv \mathcal{P}^{-1}(\mathbf{z}^i)$, it becomes apparent that (4.11) actually is a first-order Taylor expansion of the inverse mapping operator. Substituting $\mathbf{z} = \mathbf{z}^*$ into (4.11) finally yields

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \left[\frac{\partial \mathcal{P}}{\partial \mathbf{x}}(\mathbf{x}^i)\right]^{-1} \left(\mathbf{z}^* - \mathbf{z}^i\right), \tag{4.12}$$

where $\mathbf{x}^{i+1}$ is a new estimate for $\mathbf{x}^*$.

The ASM update (4.12) is reminiscent of the quasi-Newton update (2.22). The main difference is that an ordinary quasi-Newton coupling algorithm solves the root-finding problem $\mathcal{R}(\mathbf{x}) = \mathbf{0}$ at each time step, whereas an ASM coupling algorithm solves the root-finding problem $\mathbf{z}^* - \mathcal{P}(\mathbf{x}) = \mathbf{0}$ at each time step. The same quasi-Newton algorithms used to approximate the inverse-Jacobian of the residual operator are now used with a slight modification to approximate the inverse-Jacobian of the mapping operator.

At this point two loose ends remain, which are the determination of $\mathbf{z}^*$ and $\mathbf{z}^i$. The former is the converged solution of the low-fidelity FSI model at a given time level. Hence, the $\mathbf{z}^*$'s for all time levels are obtained by simulating the low-fidelity FSI model in a mono-fidelity setting using coupling algorithms from Chapter 3. The latter is a low-fidelity solution corresponding to an estimate, $\mathbf{x}^i$, for $\mathbf{x}^*$ and is obtained from an auxiliary FSI problem.

4.2.3 Auxiliary FSI problem

The iterative process of repeatedly applying (4.12) forms the main FSI problem also known as the "outer problem" and is almost similar to a mono-fidelity FSI simulation. To perform an update, however, $\mathbf{z}^i$ is required. The low-fidelity estimate is obtained from its corresponding high-fidelity estimate by evaluating $\mathcal{P}(\mathbf{x}^i)$.The evaluation of $\mathcal{P}(\mathbf{x}^i)$ is called the auxiliary FSI problem, also known as the "inner problem" and has no counterpart in a mono-fidelity FSI simulation.

Based on the definition in (4.4), the inner problem can be formulated as the root-finding problem $\tilde{\mathcal{R}}(\mathbf{z}) - \mathcal{R}(\mathbf{x}) = \mathbf{0}$, where $\mathcal{R}(\mathbf{x})$ is a constant. The inner problem is solved iteratively by repeatedly applying

$$\mathbf{z}^{j+1} = \mathbf{z}^j + \left[\frac{\partial \tilde{\mathcal{R}}}{\partial \mathbf{z}}(\mathbf{z}^j)\right]^{-1} \left(\mathbf{r}^i - \tilde{\mathbf{r}}^j\right), \tag{4.13}$$

where $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$ is the high-fidelity residual of the i-th outer-iteration and $\tilde{\mathbf{r}}^j = \tilde{\mathcal{R}}(\mathbf{z}^j)$ is the low-fidelity residual of the j-th inner-iteration at the i-th outer-iteration.

4.3 Multi-fidelity accelerated quasi-Newton coupling algorithms

Aggressive Space Mapping is a multi-fidelity quasi-Newton algorithm. Several variants of ASM can be obtained by modifying the quasi-Newton coupling algorithms from Chapter 3. Due to the requirement of an auxiliary FSI problem, the ASM-accelerated algorithms come in pairs: one for the outer iterations and one for the inner iterations.

4.3.1 ASM-GS

The Aggressive-Space-Mapping accelerated Gauss-Seidel temporal coupling-algorithm (ASM-GS) is easily derived from Algorithm 2 by comparing (2.22) and (4.12). In both equations an inverse-Jacobian is multiplied by a vector. Where in Gauss-Seidel the vector is $\mathbf{0} - \mathbf{r}^i$, in ASM-GS the vector should be $\mathbf{z}^* - \mathbf{z}^i$. The outer iterations are performed using Algorithm 7.

Algorithm 7 ASM-Gauss-Seidel (ASM-GS)

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$
1: $i = 0$
2: $\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
3: **if** $||\mathbf{r}^0|| \leq \epsilon$ **then**
4: **return** $\mathbf{x}^0$
5: **else**
6: $\mathbf{z}^0 = \texttt{ASM-MAP}(\mathbf{z}^*, \mathbf{r}^0)$
7: $\mathbf{x}^1 = \mathbf{x}^0 + (\mathbf{z}^* - \mathbf{z}^0)$
8: **end if**
9: **while** $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
10: $i = i + 1$
11: $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
12: $\mathbf{z}^i = \texttt{ASM-MAP}(\mathbf{z}^{i-1}, \mathbf{r}^i)$
13: $\mathbf{x}^{i+1} = \mathbf{x}^i + (\mathbf{z}^* - \mathbf{z}^i)$
14: **end while**
15: **return** $\mathbf{x}^{i+1}$

Algorithm 7 calls another algorithm to perform the inner iterations. The implementation of inner iterations based on Gauss-Seidel is given by Algorithm 8. Basically, Algorithm 8 is used to solve the auxiliary FSI problem using Gauss-Seidel iterations.

Algorithm 8 ASM-MAP (based on Gauss-Seidel)

Require: $\mathbf{z}^0$, $\mathbf{r}^i$, ϵ_{map}, $j_{\max}$
$j = 0$
$\tilde{\mathbf{r}}^0 = \tilde{\mathcal{R}}(\mathbf{z}^0)$
if $\|\mathbf{r}^i - \tilde{\mathbf{r}}^0\| \leq \epsilon_{\text{map}}$ **then**
  return $\mathbf{z}^0$
else
  $\mathbf{z}^1 = \mathbf{z}^0 + (\mathbf{r}^i - \tilde{\mathbf{r}}^0)$
end if
while $\|\mathbf{r}^i - \tilde{\mathbf{r}}^j\| > \epsilon_{\text{map}}$ **and** $j + 1 < j_{\max}$ **do**
  $j = j + 1$
  $\tilde{\mathbf{r}}^j = \tilde{\mathcal{R}}(\mathbf{z}^j)$
  $\mathbf{z}^{j+1} = \mathbf{z}^j + (\mathbf{r}^i - \tilde{\mathbf{r}}^j)$
end while
return $\mathbf{z}^{j+1}$

4.3.2 ASM-Aitken

As the Aitken algorithm was easily derived from the Gauss-Seidel algorithm, the Aggressive-Space-Mapping accelerated Aitken coupling algorithm (ASM-Aitken) is easily derived from the ASM-GS algorithm. The single modification is, as before, the inclusion of an adaptive relaxation parameter. The implementation of ASM-Aitken is given by Algorithm 9.

Algorithm 9 ASM-Aitken $\boldsymbol{\Delta}^2$ underrelaxation (ASM-AITKEN)

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$, ω^0
$i = 0$
$\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
if $||\mathbf{r}^0|| \leq \epsilon$ **then**
  return $\mathbf{x}^0$
else
  $\mathbf{z}^0 = \texttt{ASM-MAP}(\mathbf{z}^*, \mathbf{r}^0)$
  $\mathbf{x}^1 = \mathbf{x}^0 + \omega^0(\mathbf{z}^* - \mathbf{z}^0)$
end if
while $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
  $i = i + 1$
  $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
  $\mathbf{z}^i = \texttt{ASM-MAP}(\mathbf{z}^{i-1}, \mathbf{r}^i)$
  $\Delta\mathbf{z}^i = \mathbf{z}^i - \mathbf{z}^{i-1}$
  $\omega^i = \omega^{i-1} \cdot \dfrac{(\Delta\mathbf{z}^i)^\mathsf{T}\mathbf{z}^{i-1}}{(\Delta\mathbf{z}^i)^\mathsf{T}\Delta\mathbf{z}^i}$
  $\mathbf{x}^{i+1} = \mathbf{x}^i + \omega^i(\mathbf{z}^* - \mathbf{z}^i)$
end while
return $\mathbf{x}^{i+1}$

Inner iterations based on Aitken Δ^2 underrelaxation are performed by Algorithm 10.

Algorithm 10 ASM-MAP (based on Aitken Δ^2 underrelaxation)

Require: $\mathbf{z}^0$, $\mathbf{r}^i$, ϵ_{map}, $j_{\max}$, ω^0
$j = 0$
$\tilde{\mathbf{r}}^0 = \tilde{\mathcal{R}}(\mathbf{z}^0)$
if $\|\mathbf{r}^i - \tilde{\mathbf{r}}^0\| \leq \epsilon_{\text{map}}$ **then**
  return $\mathbf{z}^0$
else
  $\mathbf{z}^1 = \mathbf{z}^0 + \omega^0(\mathbf{r}^i - \tilde{\mathbf{r}}^0)$
end if
while $\|\mathbf{r}^i - \tilde{\mathbf{r}}^j\| > \epsilon_{\text{map}}$ **and** $j + 1 < j_{\max}$ **do**
  $j = j + 1$
  $\tilde{\mathbf{r}}^j = \tilde{\mathcal{R}}(\mathbf{z}^j)$
  $\Delta\tilde{\mathbf{r}}^j = \tilde{\mathbf{r}}^j - \tilde{\mathbf{r}}^{j-1}$
  $\omega^j = \omega^{j-1} \cdot \dfrac{(\Delta\tilde{\mathbf{r}}^j)^\mathsf{T}\tilde{\mathbf{r}}^{j-1}}{(\Delta\tilde{\mathbf{r}}^j)^\mathsf{T}\Delta\tilde{\mathbf{r}}^j}$
  $\mathbf{z}^{j+1} = \mathbf{z}^j + \omega^j(\mathbf{r}^i - \tilde{\mathbf{r}}^j)$
end while
return $\mathbf{z}^{j+1}$

4.3.3 ASM-Broyden

The Aggressive-Space-Mapping-accelerated Broyden algorithm is obtained from the mono-fidelity Broyden algorithm by replacing the changes in the residual (i.e $\Delta\mathbf{r}^i$) with changes in the low-fidelity solution (i.e. $\Delta\mathbf{z}^i$). The implementation of ASM-BROYDEN is given by Algorithm 11.

Algorithm 11 ASM-Broyden (ASM-BROYDEN)

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$
1: $i = 0$
2: $\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
3: **if** $||\mathbf{r}^0|| \leq \epsilon$ **then**
4: **return** $\mathbf{x}^0$
5: **else**
6: $J_0^{-1} = I$
7: $\mathbf{z}^0 = \texttt{ASM} - \texttt{MAP}(\mathbf{z}^*, \mathbf{r}^0)$
8: $\mathbf{x}^1 = \mathbf{x}^0 + J_0^{-1}(\mathbf{z}^* - \mathbf{z}^0)$
9: **end if**
10: **while** $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
11: $i = i + 1$
12: $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
13: $\mathbf{z}^i = \texttt{ASM} - \texttt{MAP}(\mathbf{z}^{i-1}, \mathbf{r}^i)$
14: $\Delta\mathbf{z}^i = \mathbf{z}^i - \mathbf{z}^{i-1}$
15: $\Delta\mathbf{x}^i = \mathbf{x}^i - \mathbf{x}^{i-1}$
16: $J_i^{-1} = J_{i-1}^{-1} + \dfrac{\Delta\mathbf{x}^i - J_{i-1}^{-1}\Delta\mathbf{z}^i}{(\Delta\mathbf{x}^i)^\mathsf{T} J_{i-1}^{-1}\Delta\mathbf{z}^i}(\Delta\mathbf{x}^i)^\mathsf{T} J_{i-1}^{-1}$
17: $\mathbf{x}^{i+1} = \mathbf{x}^i + J_i^{-1}(\mathbf{z}^* - \mathbf{z}^i)$
18: **end while**
19: **return** $\mathbf{x}^{i+1}$

Inner iterations based on Broyden are performed by Algorithm 12.

Algorithm 12 ASM-MAP (based on Broyden)

Require: $\mathbf{z}^0$, $\mathbf{r}^i$, ϵ_{map}, $j_{\max}$
1: $j = 0$
2: $\tilde{\mathbf{r}}^0 = \tilde{\mathcal{R}}(\mathbf{z}^0)$
3: **if** $||\mathbf{r}^i - \tilde{\mathbf{r}}^0|| \leq \epsilon_{\text{map}}$ **then**
4: **return** $\mathbf{z}^0$
5: **else**
6: $J_0^{-1} = -I$
7: $\mathbf{z}^1 = \mathbf{z}^0 - J_0^{-1}(\mathbf{r}^i - \tilde{\mathbf{r}}^0)$
8: **end if**
9: **while** $||\mathbf{r}^i - \tilde{\mathbf{r}}^j|| > \epsilon_{\text{map}}$ **and** $j + 1 < j_{\max}$ **do**
10: $j = j + 1$
11: $\tilde{\mathbf{r}}^j = \tilde{\mathcal{R}}(\mathbf{z}^j)$
12: $\Delta\tilde{\mathbf{r}}^j = \tilde{\mathbf{r}}^j - \tilde{\mathbf{r}}^{j-1}$
13: $\Delta\mathbf{z}^j = \mathbf{z}^j - \mathbf{z}^{j-1}$
14: $J_j^{-1} = J_{j-1}^{-1} + \dfrac{\Delta\mathbf{z}^j - J_{j-1}^{-1}\Delta\tilde{\mathbf{r}}^j}{(\Delta\tilde{\mathbf{z}}^j)^\mathsf{T} J_{j-1}^{-1}\Delta\tilde{\mathbf{r}}^j}(\Delta\mathbf{z}^j)^\mathsf{T} J_{j-1}^{-1}$
15: $\mathbf{z}^{j+1} = \mathbf{z}^j - J_j^{-1}\tilde{\mathbf{r}}^j$
16: **end while**
17: **return** $\mathbf{z}^{j+1}$

4.3.4 ASM-ILS

The Aggressive-Space-Mapping-accelerated IQN-ILS algorithm is obtained from the mono-fidelity IQN-ILS algorithm by replacing the changes in the residual (i.e $\Delta\mathbf{r}^i$) with changes in the low-fidelity solution (i.e. $\Delta\mathbf{z}^i$). Hence the matrix V now contains Δz^i as columns instead of $\Delta\mathbf{r}^i$. The implementation of ASM-ILS is given by Algorithm 13.

Algorithm 13 ASM-ILS

Require: $\mathbf{x}^0$, ϵ, $i_{\max}$
$i = 0$
$\mathbf{r}^0 = \mathcal{R}(\mathbf{x}^0)$
if $||\mathbf{r}^0|| \leq \epsilon$ **then**
  return $\mathbf{x}^0$
else
  $\mathbf{z}^0 = \mathtt{ASM-MAP}(\mathbf{z}^*, \mathbf{r}^0)$
  $\tilde{\mathbf{x}}^1 = \mathbf{r}^0 + \mathbf{x}^0$
  $\mathbf{x}^1 = \mathbf{x}^0 + (\mathbf{z}^* - \mathbf{z}^0)$
end if
while $||\mathbf{r}^i|| > \epsilon$ **and** $i + 1 < i_{\max}$ **do**
  $i = i + 1$
  $\mathbf{r}^i = \mathcal{R}(\mathbf{x}^i)$
  $\mathbf{z}^i = \mathtt{ASM-MAP}(\mathbf{z}^{i-1}, \mathbf{r}^i)$
  $\tilde{\mathbf{x}}^{i+1} = \mathbf{r}^i + \mathbf{x}^i$
  $\Delta\mathbf{z}^i = \mathbf{z}^i - \mathbf{z}^0$
  $\Delta\tilde{\mathbf{x}}^{i+1} = \tilde{\mathbf{x}}^{i+1} - \tilde{\mathbf{x}}^1$
  if $i == 1$ **then**
    $V = [\Delta\mathbf{z}^i]$
    $W = [\Delta\tilde{\mathbf{x}}^{i+1}]$
  else
    $V = [\Delta\mathbf{z}^i \quad V]$
    $W = [\Delta\tilde{\mathbf{x}}^{i+1} \quad W]$
  end if
  $Q, R = \mathsf{qr}(V)$
  solve $R\mathbf{c} = Q^{\mathsf{T}}(\mathbf{z}^* - \mathbf{z}^i)$
  $\mathbf{x}^{i+1} = \mathbf{x}^i + W\mathbf{c} + \mathbf{r}^i$
end while
return $\mathbf{x}^{i+1}$

Inner iterations based on IQN-ILS are performed by Algorithm 14.

Algorithm 14 IQN-ILS

Require: $\mathbf{z}^0$, $\mathbf{r}^i$, ϵ_{map}, $j_{\max}$

$j = 0$
$\tilde{\mathbf{r}}^0 = \mathcal{R}(\mathbf{z}^0)$
if $||\mathbf{r}^i - \tilde{\mathbf{r}}^0|| \leq \epsilon_{\text{map}}$ **then**
  return $\mathbf{z}^0$
else
  $\tilde{\mathbf{z}}^1 = \tilde{\mathbf{r}}^0 + \mathbf{z}^0$
  $\mathbf{z}^1 = \mathbf{z}^0 + (\mathbf{r}^i - \tilde{\mathbf{r}}^0)$
end if
while $||\mathbf{r}^i - \tilde{\mathbf{r}}^j|| > \epsilon_{\text{map}}$ **and** $j + 1 < j_{\max}$ **do**
  $j = j + 1$
  $\tilde{\mathbf{r}}^j = \tilde{\mathcal{R}}(\mathbf{z}^j)$
  $\tilde{\mathbf{z}}^{j+1} = \tilde{\mathbf{r}}^j + \mathbf{z}^j$
  $\Delta\tilde{\mathbf{r}}^j = \tilde{\mathbf{r}}^j - \tilde{\mathbf{r}}^0$
  $\Delta\tilde{\mathbf{z}}^{j+1} = \tilde{\mathbf{z}}^{j+1} - \tilde{\mathbf{z}}^j$
  if $j == 1$ **then**
    $V = \left[\Delta\tilde{\mathbf{r}}^j\right]$
    $W = \left[\Delta\tilde{\mathbf{z}}^{j+1}\right]$
  else
    $V = \left[\Delta\tilde{\mathbf{r}}^j \quad V\right]$
    $W = \left[\Delta\tilde{\mathbf{z}}^{i+1} \quad W\right]$
  end if
  $Q, R = \mathsf{qr}(V)$
  solve $R\mathbf{c} = -Q^{\mathsf{T}}\tilde{\mathbf{r}}^j$
  $\mathbf{z}^{j+1} = \mathbf{z}^j + W\mathbf{c} + \tilde{\mathbf{r}}^j$
end while
return $\mathbf{z}^{j+1}$

Chapter 5

Design philosophy and structure of CASMIR

Introduction

CASMIR (**C**oupling **A**lgorithms for **S**trongly-coupled **M**ultiphysical **I**nteraction **R**esearch) is a standalone coupling framework for the partitioned simulation of multiphysical problems using black-box solvers. It is developed as part of this thesis to provide a flexible and easily extensible tool to accelerate the set up of future partitioned fluid-structure interaction simulations. CASMIR is written in MATLAB's native M-code to enable fast development.

CASMIR incorporates built-in coupling algorithms to couple interacting physical domains in time. These temporal coupling algorithms were previously discussed at length for mono-fidelity simulations in Chapter 3 and for multi-fidelity simulations in Chapter 4. Spatial coupling in case of non-matching meshes (see Section 2.3) and data communication between CASMIR and black-box solvers is provided by FLECS.

FLECS is a flexible coupling shell written in C and designed as an interface for multidisciplinary simulations. FLECS is basically a MPI server using high-level functions based on `MPI_Send` and `MPI_Receive` to establish data transfers between connected clients.

Functions defined in the library of FLECS (written in C) are made available to CASMIR (written in M-code) through MATLAB's MEX interface used to dynamically load C, C++ or Fortran code in MATLAB and GNU Octave. CASMIR employs multiple FLECS servers (one server per black-box solver used in a simulation) to receive data from one black-box solver, manipulate the data and send (un)modified data to another black-box solver. For a black-box solver to be capable of communicating with CASMIR through FLECS, it must be linked against both MPI and FLECS. Due to the trend of parallel computing in recent decades, many existing software packages already include MPI. Hence, in most cases, only the small FLECS library should be additionally included.

CASMIR's source code is freely available under a LGPL license. The source code of FLECS is also open-source under a GPL license. Hence the complete framework is open-source and freely available. However, CASMIR has been developed and tested using MATLAB only, which is a commercial proprietary software package. Although GNU Octave is an open-source clone of MATLAB, it is not guaranteed that CASMIR will function properly when using GNU Octave instead of MATLAB.

An overview of the code structure of CASMIR is presented in Section 5.1. Configuration files used to define a partitioned multiphysics simulation using black-box solvers in CASMIR are elaborated in detail in Section 5.2.

5.1 Code structure overview

CASMIR consists out of a multitude of M-files. These files can be roughly divided into three groups:

1. Implementation of temporal coupling algorithms
2. Wrappers around FLECS functions
3. Program initialisation and work flow

The implementation of the temporal coupling algorithms is carried out in a solver-agnostic manner. Moreover, the implementation shows no indication of being geared towards a certain type of multiphysical interaction. Even though CASMIR is primarily developed for partitioned FSI simulations, the coupling algorithms have a generic setup allowing for the partitioned simulation of other multiphysics problems. This is made possible by having each coupling algorithm accept a MATLAB function as input argument which is used to evaluate the interface residual. The input function, hence, assumes the role of an interface residual operator as presented in Chapter 3 and Chapter 4. The input function is programmed to execute several data transfers of interface quantities between CASMIR and external black-box solvers and returns the interface residual computed as a function of data received by CASMIR.

FLECS basically consists of wrappers around MPI functions to simplify the transfer of field data (stored in 1-D arrays) between connected clients. Interfacial field data are bound to Euclidean coordinates (1-D, 2-D or 3-D) of the multiphysical interface. When the coordinates of the interface defined on the receiving side does not match the coordinates of the interface defined on the sending side, FLECS automatically interpolates the data transmitted by the sender to properly define the data on the interfacial nodes of the receiver. CASMIR is programmed to use FLECS in such a way that CASMIR receives unmodified data (i.e. as if interpolation is disabled) and to only interpolate data when CASMIR is sending data to an external black-box solver. The former is achieved by having CASMIR mirroring the opposing interface nodes when receiving data. This way, CASMIR can work directly on the output of black-box solvers instead of on data that has been modified by FLECS through interpolation. To simplify the further development of CASMIR a minimal set of FLECS functions have been wrapped, defining functions for CASMIR to receive interface data, send interface data and to update interface coordinates.

CASMIR is initialised through a set of configurations stored for simplicity in M-files. These M-files define things like the coupling algorithm used to reduce the interface residual, the maximum number of coupling iterations, number of stages in the time schemes used by the external black-box solvers and, most importantly, fully define the external black-box solvers in terms of input and output quantities and the physical domain they belong to. All settings defined in the configuration files are discussed in full detail in Section 5.2.

Mono-fidelity framework

In the mono-fidelity setting, two FLECS servers are used to link CASMIR ($\mathcal{C}$) to a single flow solver ($\mathcal{F}$) and a single structure solver ($\mathcal{S}$). The layout is visualised in Figure 5.1.

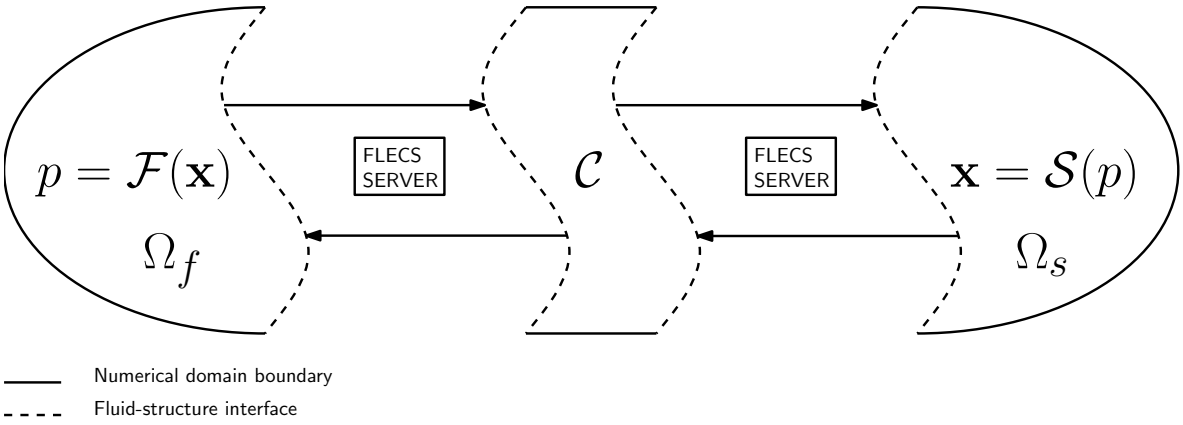


Figure 5.1: General overview of the mono-fidelity CASMIR layout

The sequence of operations performed during a time step in the mono-fidelity setting is depicted by the flowchart in Figure 5.2.

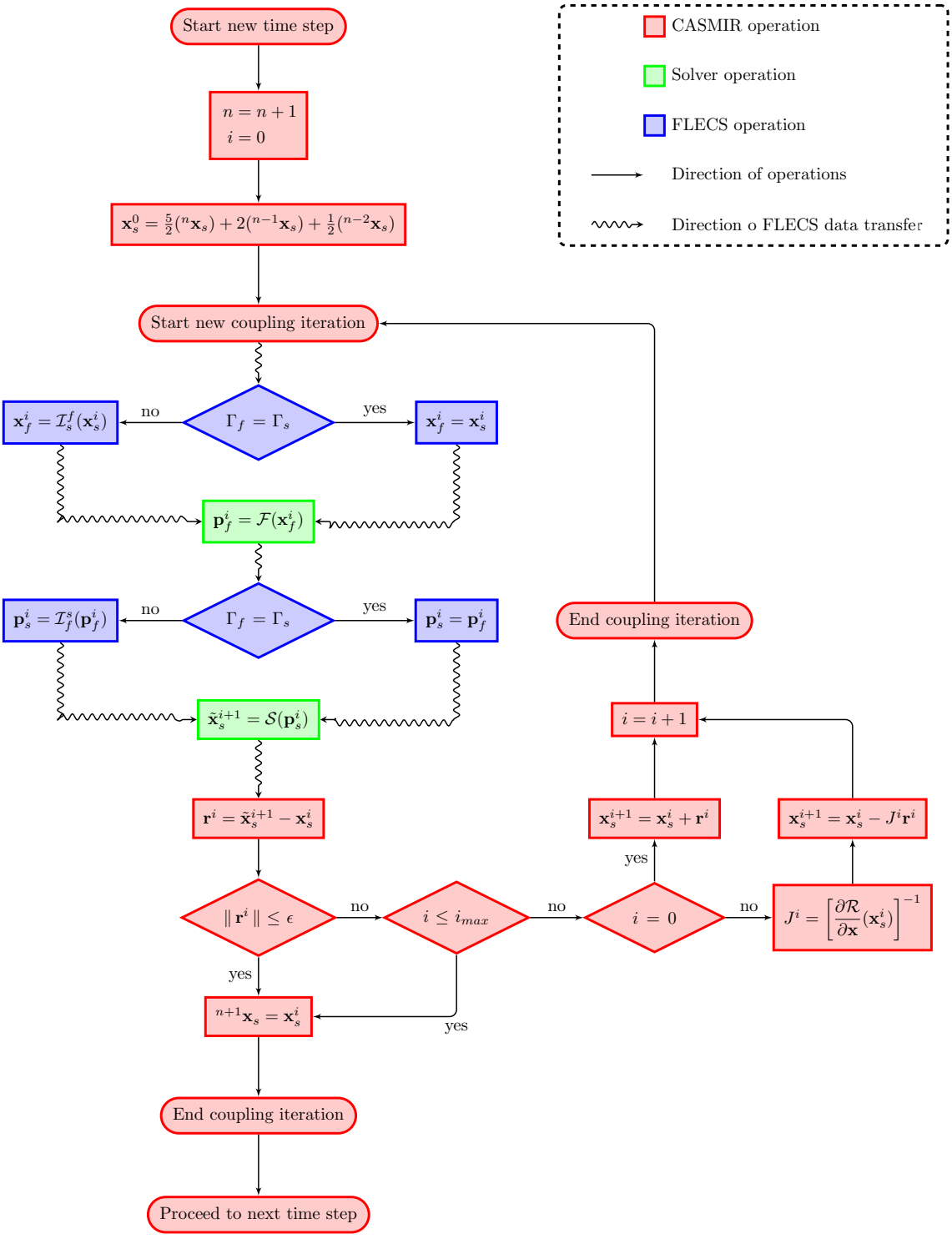


Figure 5.2: Flowchart of the mono-fidelity partitioned coupling procedure in CASMIR

Multi-fidelity framework

In the multi-fidelity setting, three FLECS servers are used to link CASMIR ($\mathcal{C}$) to a high-fidelity flow solver ($\mathcal{F}$), a low-fidelity flow solver ($\tilde{\mathcal{F}}$) and a single structure solver ($\mathcal{S}$). The layout is visualised in Figure 5.3.

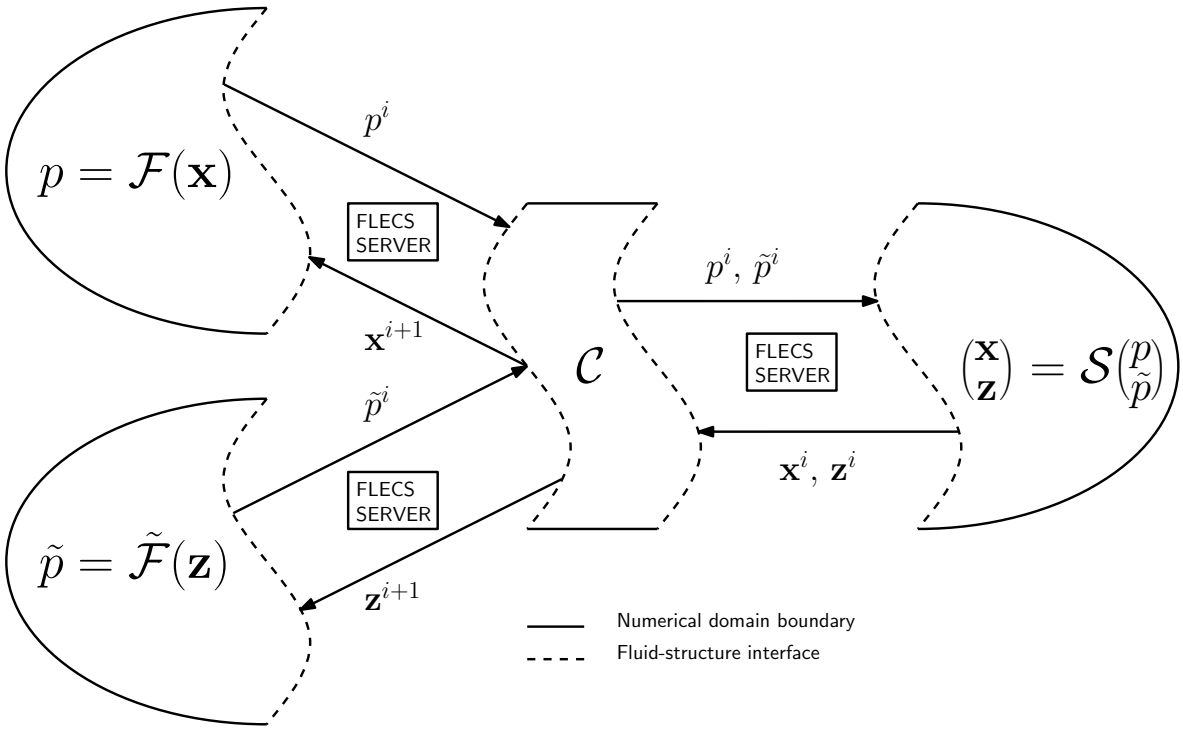


Figure 5.3: General overview of the multi-fidelity CASMIR layout

The sequence of operations performed during a single time step in the multi-fidelity setting is depicted by the flowchart in Figure 5.4.

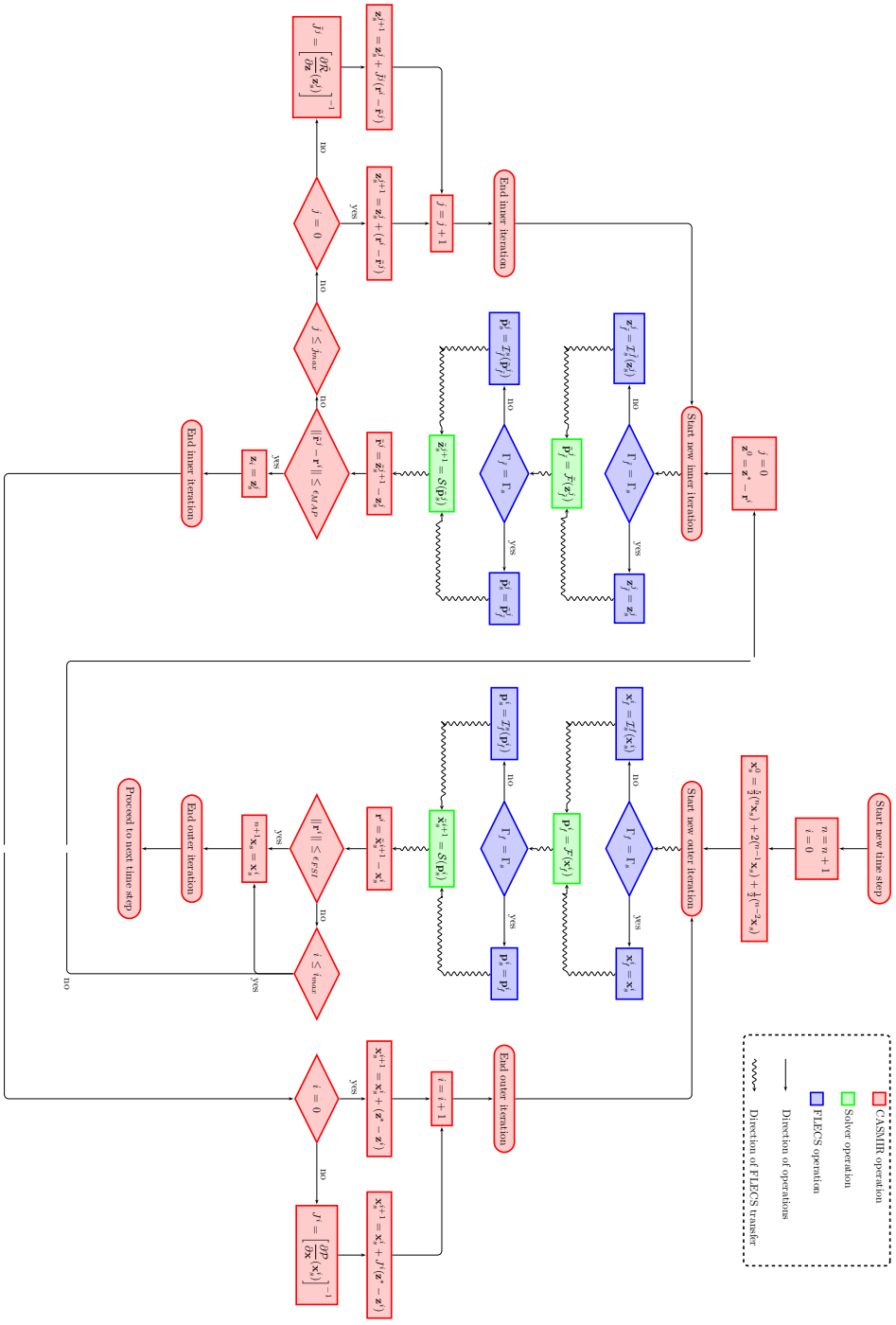


Figure 5.4: Flowchart of the multi-fidelity partitioned coupling procedure in CASMIR

5.2 Configuration files

CASMIR is configured through a set of M-files. All settings stored in the configuration M-files are specified one by one in the following tables.

physics.m

DOMAINS	1-D cell array of strings listing the names of the physical domains involved in the simulation.
FIELDS	1-D cell array of strings listing the names of the physical fields used as inputs and outputs by the external black-box solvers.
FIELD_ACTIVE_DIM	1-D cell array of 1-D arrays defining the components of vector fields. Use [0, 0, 0] in case of a scalar field.

solver_n.m

SOLVER_NAME	String specifying the name of solver n.
DOMAIN_ID	Integer referring to one of the domains specified in DOMAINS.
INPUT_ID	Array of integers referring to one or several interfacial fields defined in FIELDS specifying the input fields of solver operator n.
OUTPUT_ID	Array of integers referring to one or several interfacial fields defined in FIELDS specifying the output fields of solver operator n.
FIDELITY_LEVEL	Integer specifying the level of fidelity of solver operator n. Use 1 for the highest level and consecutive numbers for lower levels.
EXE_DIR	String specifying the location of the root directory containing the 'executable' of solver n.
RUN_CMD	String specifying the command-line used in a terminal to launch solver n.

interaction.m

TRANSFER_MATRIX	1-D cell array of 1-D arrays of integers. Each array defines which scalar/vector field CASMIR receives/sends from/to a solver. Each row defines a single uni-directional scalar/vector field transfer. The colomns represent the following: (1) Use 0 if CASMIR is receiving a scalar/vector field. Use 1 if CASMIR is receiving a scalar/vector field. (2) ID of the solver involved either sending to CASMIR or receiving from CASMIR. (3) ID of the scalar/vector field that is being transferred. (4) If (1) equals 0: set value equal to (2). If (1) equals 1: set value to ID of the solver from which CASMIR originally received the scalar/vector field it is currently sending.
RESIDUAL_OPERATOR	String specifying the MATLAB function serving as the residual operator.
RESIDUAL_ARGUMENT	Array of integers specifying the IDs of the fields used as input for the residual operator.
RESIDUAL_ARGUMENT_SUPPLIER	Integer specifying the ID of the solver having the fields referred to in RESIDUAL_ARGUMENT as output.

interface.m

DYNAMIC_INTERFACE	Integer specifying whether the multiphysical interface is static or dynamic. Use 0 for static and 1 for dynamic.
COORDINATE_UPDATE_MODE	Integer specifying the mode to use when updating the coordinates of interfaces. Use 0 if the old coordinates are to be replaced by new coordinates. Use 1 if a displacement is added to the old coordinates.
COORDINATE_UPDATER_ID	Integer specifying the ID of the field used to update interface coordinates.

algorithms.m

INVERSE_JACOBIAN	String specifying the coupling algorithm to be used by CASMIR.
AITKEN_OMEGA	Float specifying the initial value of the underrelaxation parameter used by the Aitken Δ^2 coupling algorithm.
IQN_ILS_NUM_REUSE	Integer specifying the number of previous time levels to be reused by the IQN-ILS(r) algorithm (i.e. this specifies r).
IQN_ILS_DIAG_MIN_ ABS_VAL	Float specifying the tolerance for detecting small diagonal values on the QR decomposition employed by the IQN-ILS and IQN-ILS(r) algorithms.
RESIDUAL_TOLERANCE	Cell array of floats specifying the convergence tolerances starting from the highest fidelity coupling.
MAX_COUP_ITER	Cell array of integers specifying a cap on the number of coupling iterations starting from the highest fidelity coupling.

data.m

NUM_TIME_LEVELS	Integer specifying the number of time steps to be performed by the external black-box solvers.
NUM_RK_STAGES	Integer specifying the number of stages in the time scheme used by the external black-box solvers.
EXPLICIT_RK_STAGES	1-D array of integers specifying whether a stage is explicit or implicit. Use 0 is the stage is implicit and use 1 if the stage is explicit.

flecs.m

LOG_LEVEL	Integer specifying the amount of messages to be printed in the terminal where FLECS is running.
CONN_MODE	FLECS constant specifying the connection mode used by FLECS.
ERROR_MODE	FLECS constant specifying the error mode used by FLECS.
EXE_DIR	String specifying the root directory containing the FLECS executable.

Chapter 6

Description of FSI test cases

Introduction

In Chapter 5 CASMIR was introduced as a standalone coupling framework for the partitioned simulation of multiphysical problems using black-box solvers. To test the capabilities and performance of CASMIR, two FSI cases are defined in this chapter. The first case is the simulation of a flow through a radially flexible channel resembling blood flow through veigns. The second case is the simulation of a flow over a clamped flexible beam resembling the vibration of thin panels or control surfaces on a aircraft.

Several test cases based on the two FSI cases are defined in this chapter and used to assess the performance of both the mono-fidelity coupling algorithms from Chapter 3 and the multi-fidelity coupling algorithms from Chapter 4. The governing equations, numerical domain, discretisation and associated black-box solvers of the supersonic panel flutter problem are described in Section 6.1. Similarly, the channel flow problem is described in Section 6.2. Results obtained from simulations using CASMIR are collected in Chapter 7.

6.1 Supersonic panel flutter

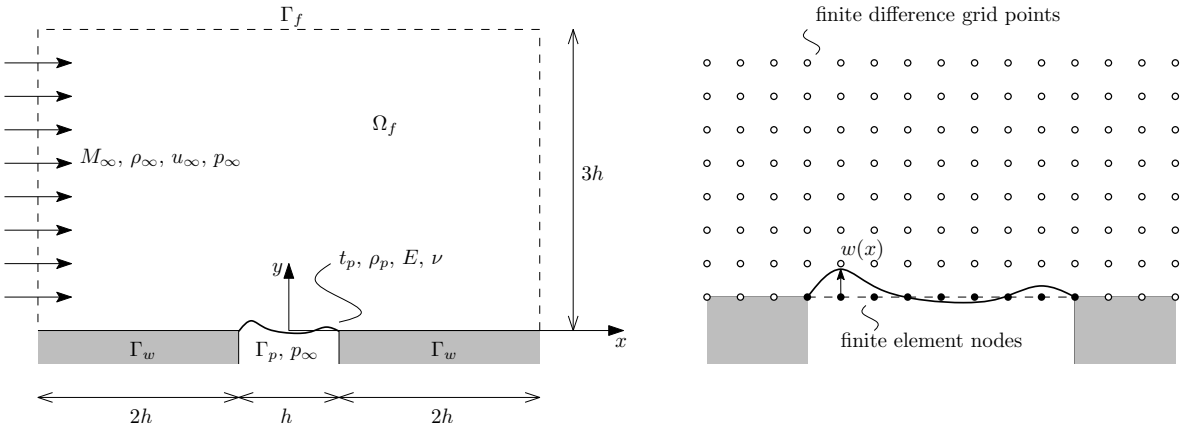


Figure 6.1: Schematic overview of fluid domain, boundaries and panel (left) and the discretisation of the domains (right) for the supersonic panel flutter problem

In the panel flutter problem as depicted in Figure 6.1, the fluid domain is rectangular. Only the bottom side of the fluid domain is bordered by a solid wall. This wall consists of two rigid portions on the left and the right with the flexible panel in between.

The panel is given an initial displacement and is allowed to deform in the y direction only. In case of no flow, the initial displacement leads to a vibration of an unforced dynamic system. A uniform inflow is prescribed on the left side of the fluid domain. This flow interacts with the panel by applying a fluid pressure on the panel and hence modifying its vibrational behaviour to that of a forced dynamic system. The vibration of the panel perturbes the flow. Considering the flow is assumed to be supersonic, only a portion of the part of the fluid domain downstream from the front of the panel is perturbed.

Parameters used in Figure 6.1 related to the fluid and the corresponding flow are clarified in Table 6.1. Likewise the panel properties and the boundary/domain symbols are clarified in Table 6.2 and Table 6.3, respectively.

Table 6.1: Panel flutter - fluid properties

M_∞	Mach number
a_∞	Speed of sound
u_∞	x-dir. fluid velocity
ρ_∞	Fluid density
p_∞	Fluid pressure

Table 6.2: Panel flutter - panel properties

t_p	Panel thickness
h	Panel length
ρ_p	Panel density
E	Young's modulus
I	Mom. of inertia ($\frac{1}{12}t_p^3$)
ν	Poisson ratio
w	y-dir. displacement

Table 6.3: Domain and boundaries

Γ_f	Numer. boundary
Γ_w	Rigid boundary
Γ_p	Panel boundary
Ω_f	Fluid domain

6.1.1 High-fidelity flow solver

The fluid flowing over the flexible panel is assumed to be a compressible Newtonian fluid. The unperturbed flow is horizontal.

Governing equations

The supersonic flow over the flexible panel is governed by the two-dimensional unsteady linearised potential flow equation

$$(1 - M_\infty)\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} - \frac{1}{a_\infty^2}\left(\frac{\partial^2 \phi}{\partial t^2} + 2u_\infty \frac{\partial^2 \phi}{\partial x \partial t}\right) = 0. \tag{6.1}$$

The potential is related to the horizontal flow velocity, u and the vertical flow velocity, v, through

$$\begin{aligned} u &= u_\infty + \frac{\partial \phi}{\partial x}, \\ v &= \frac{\partial \phi}{\partial y}. \end{aligned} \tag{6.2}$$

The linearised potential is only valid for subsonic and supersonic flows, yet invalid in the transonic regime.

Discretisation

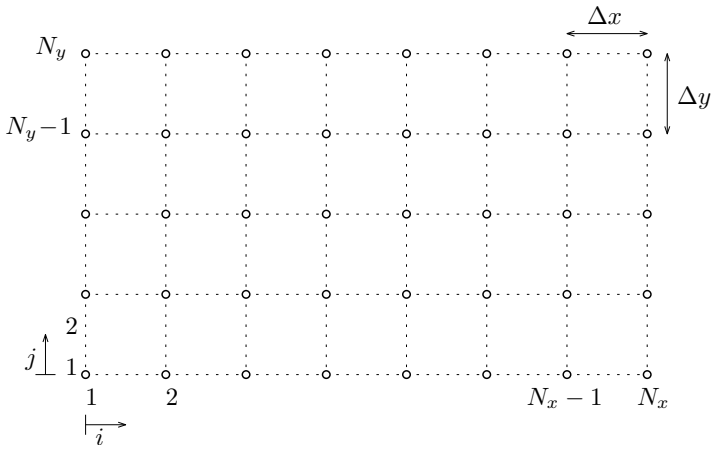


Figure 6.2: Fluid domain grid of size $N_x \times N_y$

The linearised potential flow equation is discretised using finite differences. An upwind discretisation scheme is applied in order to be consistent with the domain of influence of a supersonic flow. Pertubations propagate only downstream in a supersonic flow, hence the discretisation of the potential at a given point can only depend on discrete values that lie upstream with respect to the given point. The upwind finite difference discretisation of the terms in (6.1) are collected in Table 6.4. The use of indices and grid spacings are clarified in Figure 6.2.

Table 6.4: First-order upwind finite difference discretisation of the linearised potential flow equation using indices and spacings as defined in Figure 6.2

Term	Discretisation
$\phi(x_i,\, y_j)$	$\phi_{i,j}$
$\frac{\partial \phi}{\partial x}(x_i,\, y_j)$	$\frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x}$
$\frac{\partial \phi}{\partial x}(x_1,\, y_j)$	$\frac{\phi_{1,j}}{\Delta x}$
$\frac{\partial \phi}{\partial x}(x_{N_x},\, y_j)$	$\frac{\phi_{N_x,j} - \phi_{N_x-1,j}}{\Delta x}$
$\frac{\partial^2 \phi}{\partial x^2}(x_i,\, y_j)$	$\frac{\phi_{i-2,j} - 2\phi_{i-1,j} + \phi_{i,j}}{\Delta x^2}$
$\frac{\partial^2 \phi}{\partial x^2}(x_1,\, y_j)$	$\frac{\phi_{1,j}}{\Delta x^2}$
$\frac{\partial^2 \phi}{\partial x^2}(x_2,\, y_j)$	$-2\frac{\phi_{i,j} + \phi_{2,j}}{\Delta x^2}$
$\frac{\partial^2 \phi}{\partial x^2}(x_{N_x},\, y_j)$	$\frac{\phi_{N_x-2,j} - 2\phi_{N_x-1,j} + \phi_{N_x,j}}{\Delta x^2}$
$\frac{\partial^2 \phi}{\partial y^2}(x_i,\, y_j)$	$\frac{\phi_{i,j-1} - 2\phi_{i-1,j} + \phi_{i,j+1}}{\Delta y^2}$
$\frac{\partial^2 \phi}{\partial y^2}(x_i,\, y_1)$	$\frac{2\phi_{i,2} - 2\phi_{i,1}}{\Delta y^2} - \frac{2v(x_i, y_1)}{\Delta y}$
$\frac{\partial^2 \phi}{\partial y^2}(x_i,\, y_{N_y})$	$\frac{\phi_{i,N_y-1} - 2\phi_{i,N_y}}{\Delta y^2}$

Substituting the discretisations from Table 6.4 into (6.1) results in a system of linear algebraic equations of the form

$$\mathbf{M}^{ff}\ddot{\boldsymbol{\phi}} + \mathbf{C}^{ff}\dot{\boldsymbol{\phi}} + \mathbf{K}^{ff}\boldsymbol{\phi} = \mathbf{f}^{f}, \tag{6.3}$$

where the vector $\boldsymbol{\phi}$ contains the discrete values of the potential and the vector $\mathbf{f}^f$ contains the forcing from the panel on the flow stemming from boundary conditions.

Boundary conditions

The boundary conditions (b.c.'s) for the linearised potential flow equations are

$$\begin{aligned} &\phi(x,y) = 0 && \text{on } \Gamma_f, \\ &v(x,y) \equiv \frac{\partial \phi}{\partial y} = 0 && \text{on } \Gamma_w, \\ &v(x,y) = \frac{Dw}{Dt} \equiv \frac{\partial w}{\partial t} + M_\infty a_\infty \frac{\partial w}{\partial x} && \text{on } \Gamma_p. \end{aligned} \tag{6.4}$$

The first b.c. simply states that the potential is equal to zero along the numerical boundary. The second and the third b.c.'s state that the vertical flow velocity at the bottom of the fluid domain matches that of the bordering solid boundaries. In the middle, where the panel is located, the vertical flow velocity equals the time-rate of the vertical displacement of the panel. Along the stationary rigid walls left and rigth from the panel, the vertical flow velocity is simply zero.

Discretising the displacement, w, in (6.4) leads to the forcing, $\mathbf{f}^f$, of equation (6.3) expressed as a system of linear conditions

$$\mathbf{f}^f = -\mathbf{C}^{fp}\dot{\mathbf{q}} - \mathbf{K}^{fp}\mathbf{q}, \tag{6.5}$$

where $\mathbf{q}$ and $\dot{\mathbf{q}}$ contain, respectively, the discrete values of the generalised displacements and velocities of the panel.

6.1.2 Low-fidelity flow solver

A low-fidelity fluid model is obtained from the piston analogy [27]. This leads to an expression for the pressure difference across the panel as function of the state of the panel:

$$\Delta p = \rho_\infty M_\infty a_\infty \left(\frac{M_\infty^2 - 2}{\sqrt{(M_\infty^2 - 1)^3}} \frac{\partial w}{\partial t} + M_\infty a_\infty \frac{\partial w}{\partial x} \right), \tag{6.6}$$

with $x \in \Gamma_p$. The piston analogy is valid for $M_\infty > 1.6$ only.

6.1.3 Structure solver

The flexible panel is modeled as a clamped beam using Euler-Bernoulli beam theory to describe its dynamical behaviour.

Governing equations

The dynamical behaviour of the panel is governed by the Euler-Bernoulli equation. Using the definitions in Table 6.2, gives

$$\rho_p t_p \frac{\partial^2 w}{\partial t^2} + \frac{\partial^2}{\partial x^2}\left(\frac{EI}{1-\nu^2} \frac{\partial^2 w}{\partial x^2} \right) = -\Delta p, \tag{6.7}$$

where Δp is the pressure difference across the panel.

Discretisation

Equation (6.7) is discretised using finite elements. Multiplying by a weigth function, γ and integrating twice by parts over an element results in

$$\int_{-h/2}^{h/2} \rho_p t_p \frac{\partial^2 w}{\partial t^2} \gamma \mathrm{d}x + \int_{-h/2}^{h/2} \frac{EI}{1-\nu^2} \frac{\partial^2 w}{\partial x^2} \frac{\partial^2 \gamma}{\partial x^2} \mathrm{d}x = \int_{-h/2}^{h/2} -\Delta p \gamma \mathrm{d}x. \tag{6.8}$$

Due to second-order spatial derivates of the panel displacement, both the panel displacement and slope must be continuous across inter-element boundaries. This implies that Lagrangian shape functions

cannot be used. Hermitian shape functions are applied instead (see Appendix A for a derivation of Hermitian shape functions).

Using Hermitian shape functions ψ_i the panel displacement, w, of an arbitrary element is approximated as

$$w(x,t) \approx w^h(x,t) \equiv \sum_{i=1}^{4} \psi_i(x) q_i(t), \tag{6.9}$$

where $\mathbf{q}$ contains the nodal displacements and the nodal slopes. Finally, employing a Bubnov-Galerkin method (where the same polynomials are used to represent both the shape functions and the weigth functions) yields a system of linear algebraic equations,

$$\mathbf{M}^{pp}\ddot{\mathbf{q}} + \mathbf{K}^{pp}\mathbf{q} = \mathbf{f}^p, \tag{6.10}$$

where

$$M_{ij}^{pp} = \rho_p t_p \int_{-h/2}^{h/2} \psi_i \psi_j \mathrm{d}x, \tag{6.11}$$

$$K_{ij}^{pp} = \frac{EI}{1-\nu^2} \int_{-h/2}^{h/2} \frac{\partial^2 \psi_i}{\partial x^2} \frac{\partial^2 \psi_j}{\partial y^2} \mathrm{d}x, \tag{6.12}$$

$$f_i^p = -\int_{-h/2}^{h/2} \Delta p \psi_i \mathrm{d}x, \tag{6.13}$$

for i, $j = 1$, ..., n_p. The pressure difference, Δp, in the forcing $\mathbf{f}^p$ is computed as

$$\Delta p(x) = -\rho_\infty \left(\frac{\partial \phi}{\partial t} + u_\infty \frac{\partial \phi}{\partial x} \right), \tag{6.14}$$

where ϕ is a fluid pontetial used to describe the governing flow equations. Upon discretisation of the potential, the forcing can be expressed in the form given by

$$\mathbf{f}^p = -\mathbf{C}^{pf}\dot{\boldsymbol{\phi}} - \mathbf{K}^{pf}\boldsymbol{\phi}, \tag{6.15}$$

where the vector $\boldsymbol{\phi}$ contains discrete values for the potential.

6.1.4 Simulation parameters

This section defines and lists values for all physical and numerical parameters used for simulations of the supersonic panel flutter problem.

Physical parameters

For the supersonic panel flutter problem the Mach number, M_∞, the fluid-to-structure mass ratio, ζ and the characteristic time-scale ratio, λ, are used as similarity parameters and they are defined as

$$M_\infty = \frac{u_\infty}{a_\infty}, \tag{6.16}$$

$$\zeta = \frac{\rho_\infty L}{\rho_p t_p} \tag{6.17}$$

and

$$\lambda = \frac{L a_\infty^{-1}}{(\rho_p t_p)^{1/2} L^2 (EI)^{-1/2}}. \tag{6.18}$$

In order to simulate the fluttering of the panel, the initial panel displacement is expressed as a function of the flutter mode, $\boldsymbol{\xi}$, as presented in Figure 6.3. The initial velocity of the panel is set to zero.

The initial flow field, as seen in Figure 6.4, is obtained from a steady-state solution around the initial panel displacement. The freestream Mach number is chosen to be equal to the critical Mach number, M_{cr}, obtained from linear stability analysis.

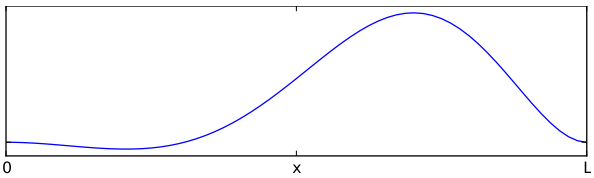


Figure 6.3: Initial panel displacement: $w^0 = 0.1\boldsymbol{\xi}/\|\boldsymbol{\xi}\|$

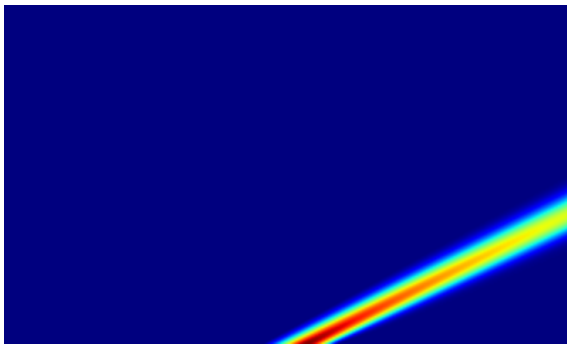


Figure 6.4: Steady-state flow around the initial panel displacement

The non-dimensional critical coupled period of the flutter mode is defined as

$$P_{cr} = \frac{2\pi a_\infty}{\omega_{cr} L}. \tag{6.19}$$

Three sets of values for all parameters are used (see Table 6.5, Table 6.6 and Table 6.7) to define a weak, medium and strong interaction between the supersonic flow and the flexible clamped panel.

Table 6.5: Similarity parameters for the supersonic panel flutter problem

Test case	M_∞ [−]	ζ [−]	λ [−]
FSI-weak	2.27	5.47E-2	1.47E-2
FSI-medium	2.28	7.41E-2	1.47E-2
FSI-strong	2.33	3.00E-1	1.47E-2

Table 6.6: Physical parameters for the supersonic panel flutter problem

Test case	$\rho_\infty \left[\frac{kg}{m^3}\right]$	$a_\infty \left[\frac{m}{s}\right]$	$\rho_p \left[\frac{kg}{m^3}\right]$	$E \left[\frac{kg}{ms^2}\right]$	$L\ [m]$	$t_p\ [m]$
FSI-weak	0.4	300	2.71E3	8.67E10	0.5	1.35E-3
FSI-medium	0.4	300	2.00E3	6.40E10	0.5	1.35E-3
FSI-strong	0.4	300	5.00E2	1.60E10	0.5	1.35E-3

Table 6.7: Critical parameters for the supersonic panel flutter problem

Test case	$M_{cr}\ [-]$	$\omega_{cr} \left[\frac{rad}{s}\right]$	$P_{cr}\ [-]$
FSI-weak	2.27	460	8.2
FSI-medium	2.28	536	7.0
FSI-strong	2.33	1075	3.5

Numerical parameters

To study the influence of the time step on the performance of the coupling algorithms a simulation time equal to P_{cr} is divided into 10, 50 and 70 intervals. To study the influence of fluid grid sizes on the performance of the coupling algorithms a small, medium and large number of fluid grid points is used. Note, however, that the fluid grids are used only by the high-fidelity solver as the low-fidelity solver does not require any discretisation. The number of finite elements used to discretise the panel is chosen such that the number of nodes equals the number of fluid grid points at the fluid-structure interface. This way the fluid mesh and the structure mesh match at the interface, hence, interpolation is superfluous. The numerical parameters are listed in Table 6.8.

Table 6.8: Numerical parameters for the supersonic panel flutter problem

Parameter	Symbol	Value
Number of time steps	N_t	$20 \cup 60 \cup 140$
Time step	Δt	$2P_{cr}/N_t$
Fluid grid: small	$N_x \times Ny$	161×97
Fluid grid: medium	$N_x \times Ny$	321×193
Fluid grid: large	$N_x \times Ny$	641×385
Number of finite elements	N_e	$32 \cup 64 \cup 128$
Number of finite element nodes	N_n	$N_e + 1$

6.2 Quasi-one-dimensional channel flow

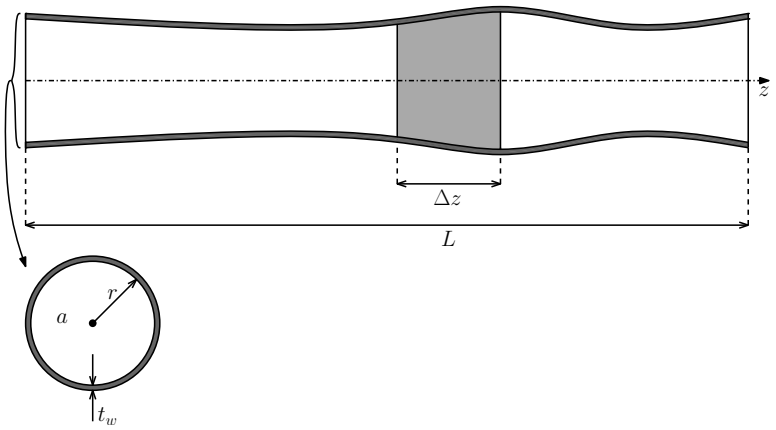


Figure 6.5: Schematic overview of the channel, cross-sectional area and geometric properties

In the channel flow problem, the structure of the channel is geometrically defined as a thin-walled circular tube (open at both ends) as depicted in Figure 6.5. Initially, the channel is at rest and undeformed with a radius of r_0. Likewise, the flow is initially at rest with a uniform axial velocity of u_0. The fluid pressure acting in the radial direction on the inner side of the channel wall at rest conditions is set to p_0.

A periodic inflow condition $u_{\text{in}}(t)$ is applied at the left opening of the channel. The fluid-pressure distribution now varies in time due to the inflow, forcing the channel to deform radially. The change in cross-sectional area of the channel further perturbes the flow through the principle of conservation of mass.

Considering that the channel is circular and has homogeneous properties and that the flow only has an axial velocity component, the channel flow problem is axis-symmetric. The problem reduces to a quasi-one-dimensional flow through a flexible channel. The numerical domain is the upper half of the channel which intersects with a plane through the centerline of the channel.

Symbols used to describe the fluid and the flow are defined in Table 6.9. Symbols used to describe the channel's geometrical and structural properties are defined in Table 6.10.

Table 6.9: Channel flow - fluid properties

ρ_f	Fluid density
u	Axial flow velocity
p	Fluid pressure

Table 6.10: Channel flow - channel properties

L	Channel length
a	Cross-sectional area
r	Inner radius
t_w	Wall thickness
E	Young's modulus
c_{MK}	Moens-Korteweg wave speed

6.2.1 High-fidelity flow solver

The fluid contained within the circular channel is assumed to be both Newtonian and incompressible. Due to the axis-symmetric nature of both the channel properties and flow conditions, only the axial component of the flow velocity is considered.

Governing equations

The flow is governed by the principles of conservation of mass and conservation of momentum, reformulated for a deforming control volume [28] yielding, respectively,

$$\frac{\partial a}{\partial t} + \frac{\partial au}{\partial z} = 0 \tag{6.20}$$

and

$$\frac{\partial au}{\partial t} + \frac{\partial au^2}{\partial z} + \frac{1}{\rho_f}\left(\frac{\partial ap}{\partial z} - p\frac{\partial a}{\partial z}\right) = 0, \tag{6.21}$$

where ρ_f is the fluid density, $a = \pi r^2$ is the cross-sectional area of the channel with radius r, u is the flow velocity along the axial direction z and p is the fluid pressure.

Discretisation

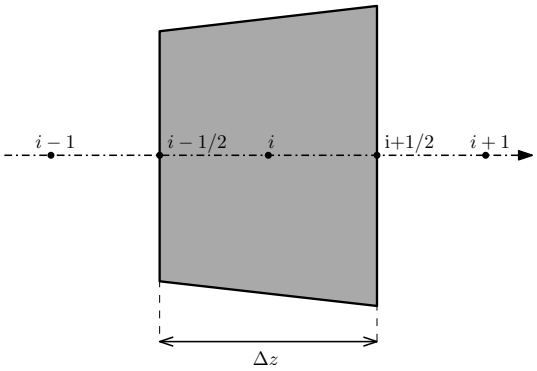


Figure 6.6: Enlargement of the shaded area of the channel in Figure 6.5, now showing the discretisation of the fluid domain.

The fluid domain is discretised with N equally-sized cells of Δz in width along the axial direction z as shown in Figure 6.6. Defining flow variables to be cell-centered, the governing equations are discretised using a first-order upwind discretisation scheme for the convective term in the momentum equation and a central discretisation scheme for all other terms. Using a Backward Euler scheme for the time derivatives (while hiding the superscript $n+1$ for the unknowns) yields

$$\frac{\Delta z}{\Delta t}(a_i - a_i^n) + a_{i+1/2}u_{i+1/2} - a_{i-1/2}u_{i-1/2} - \frac{\alpha}{\rho}(p_{i+1} - 2p_i + p_{i-1}) = 0, \tag{6.22}$$

$$\begin{aligned}\frac{\Delta z}{\Delta t}(a_i u_i - a_i^n u_i^n) + a_{i+1/2}u_{i+1/2}u_i - a_{i-1/2}u_{i-1/2}u_{i-1} \\ + \frac{1}{2\rho}(a_{i+1/2}(p_{i+1} - p_i) + a_{i-1/2}(p_i - p_{i-1})) = 0,\end{aligned} \tag{6.23}$$

where

$$\begin{aligned} u_{i-1/2} &= (u_{i-1} + u_i)/2, \\ u_{i+1/2} &= (u_i + u_{i+1})/2. \end{aligned} \tag{6.24}$$

The subscripts $i-1$, i and $i+1$ indicate cell centres and $i-1/2$ and $i+1/2$ indicate cell faces. Equation (6.22) contains a pressure stabilisation term [28] with coefficient

$$\alpha = \frac{a_0}{u_0 + \frac{\Delta z}{\Delta t}}, \tag{6.25}$$

in order to prohibit pressure wiggles due to the central discretisation in the momentum equation (6.23).

Boundary conditions

The inflow at the left opening is set by the perodic function

$$u_{in}(t) = u_0 + \frac{u_0}{10} sin^2\left(\frac{u_0 \pi n \Delta t}{L}\right). \tag{6.26}$$

The fluid pressure at the inlet is linearly extrapolated using

$$p_{in} = 2p_1 - p_2. \tag{6.27}$$

Likewise, the fluid velocity at the outlet is linearly extrapolated using

$$u_{out} = u_N - u_{N-1}. \tag{6.28}$$

The fluid pressure at the outlet is defined such that a non-reflecting boundary is ensured

$$p_{out} = 2\rho_f \left[c_{MK}^2 - \left(\sqrt{c_{MK}^2 - \frac{p_{out}^n}{2\rho_f}} - \frac{u_{out} - u_{out}^n}{4}\right)^2\right], \tag{6.29}$$

where c_{MK} is the Moens-Korteweg wave speed defined as

$$c_{MK} = \sqrt{\frac{E t_w}{2\rho_f r_0}}. \tag{6.30}$$

6.2.2 Low-fidelity flow solver

As low-fidelity model a relatively coarser discretisation of the finite-volume grid of the fluid domain will be coupled with the exact same structure model used for the high-fidelity coupling.

6.2.3 Structure solver

Assuming that the inertia of the structure can be neglected relative to the fluid inertia, the structure can be considered to be massless. Using a Hookean constitutive relation for the elastic behaviour of the channel, the circumferential stress σ_c, in the channel wall is approximated by

$$\sigma_c = E\frac{r - r_0}{r_0} + \sigma_{c_0}, \tag{6.31}$$

where E is the Young's modulus, r is the channel's inner radius and the subscript. The force balance on the fluid-structure interface is defined as

$$rp = \sigma_c t_w. \tag{6.32}$$

where t_w is the wall thickness. Substituting the constitutive relation (6.31), r_0, $p_0 = \sigma_0 t_w$ and $a = \pi r^2$ into the force balance (6.32) yields the structural deformation in terms of the cross-sectional area as function of the fluid pressure acting on the inner side of the channel wall.

$$a = a_0 \left(\frac{\frac{p_0}{2\rho_f} - c_{MK}^2}{\frac{p}{2\rho_f} - c_{MK}^2} \right)^2. \tag{6.33}$$

Although nonlinear, the structural deformation equation (6.33) is an explicit algebraic relation and thus does not require any discretisation.

6.2.4 Simulation parameters

In this section values are listed for all physical and numerical parameters used for the simulation of the channel flow problem.

Physical parameters

Two similarity parameters are identified for the channel flow problem; the dimensionless stiffness, κ, and the dimensionless time step, τ, defined as

$$\kappa = \frac{\sqrt{\frac{Et_w}{2\rho_f r} - \frac{p}{2\rho_f}}}{u_0} \tag{6.34}$$

and

$$\tau = \frac{u_0 \Delta t}{L}. \tag{6.35}$$

All simulations of the channel flow problem use the values $\kappa = 16$ and $\tau = 0.0025$. The values for the physical parameters are listed in Table 6.11.

Table 6.11: Physical parameters for the channel flow problem

Parameter	Symbol	Value
Constant fluid density	ρ_f	$1\ kg/m^3$
Initial fluid velocity	u_0	$0.1\ m/s$
Initial fluid pressure	p_0	$0\ Pa$
Initial cross-sectional area	a_0	$0.1\ m^2$
Initial inner-radius	r_0	$\sqrt{0.1/\pi}\ m$
Channel length	L	$1\ m$
Channel wall thickness	t_w	$\sqrt{\pi/4}\ m$
Young's modulus	E	$1\ Pa$

Numerical parameters

Numerical parameters used for the channel flow simulations are listed in Table 6.12.

Table 6.12: Numerical parameters for the channel flow problem

Parameter	Symbol	Value
Time step	Δt	$0.025\ s$
Number of time steps	N_t	400
Number of cells	N_v	$80 \cup 250$
Outer tolerance	ϵ_O	10^{-9}
Inner tolerance	ϵ_I	10^{-12}

6.3 Simulation designations

The performance of CASMIR is assessed using the test problems described in Section 6.2 and Section 6.1. A large number of simulations are run using different settings for the relevant physical and numerical parameters. To easily indicate which settings are used for a particular simulation all simulations are given names formatted as

`[TEST_PROBLEM_ID]-S[INTERACTION_ID]G[GRID_ID]T[STEPS_ID]`

using

$$\texttt{[TEST\_PROBLEM\_ID]} = \begin{cases} \text{CF} & \rightarrow \text{ Channel flow} \\ \text{SPF} & \rightarrow \text{ Supersonic panel flutter} \end{cases}$$

$$\texttt{[INTERACTION\_ID]} = \begin{cases} 1 & \rightarrow \text{ FSI-weak} \\ 2 & \rightarrow \text{ FSI-medium} \\ 3 & \rightarrow \text{ FSI-strong} \end{cases}$$

$$\texttt{[GRID\_ID]} = \begin{cases} 1 & \rightarrow \text{ Small grid} \\ 2 & \rightarrow \text{ Medium grid} \\ 3 & \rightarrow \text{ Large grid} \end{cases}$$

$$\texttt{[STEPS\_ID]} = \begin{cases} 1 & \rightarrow \text{ Small number of time steps} \\ 2 & \rightarrow \text{ Medium number of time steps} \\ 3 & \rightarrow \text{ Large number of time steps} \end{cases}$$

Hence, a *supersonic panel flutter* simulation of a *strong interaction* on a *small grid* using a *medium number of time steps* is designated by `SPF-S3G1T2`.

Chapter 7

Performance comparison of coupling algorithms

Results obtained from simulating the test cases defined in Chapter 6 using the mono-fidelity coupling algorithms from Chapter 3 and the multi-fidelity coupling algorithms from Chapter 4 are presented and discussed in this chapter. Results for the supersonic panel flutter problem are presented in Section 7.1. Results for the quasi-one-dimensional channel flow problem are presented in Section 7.2. Both sections follow the same format in which the mono-fidelity results are presented first and discussed after, followed by the presentation of the multi-fidelity results and concluding with the discussion of the multi-fidelity results.

7.1 Supersonic panel flutter

Results obtained from simulating the supersonic panel flutter problem using mono-fidelity coupling algorithms are presented in Section 7.1.1. Multi-fidelity results are presented in Section 7.1.2. Multi-fidelity results include the application of several algorithms for the outer iterations. For the inner iterations Gauss-Seidel is used exclusively.

7.1.1 Mono-fidelity results

Table 7.1 through Table 7.4 report the average number of coupling iterations it takes Gauss Seidel, Aitken, Broyden and IQN-ILS(0), respectively, to converge the supersonic panel flutter problem up to an interface residual norm of $\|\mathbf{r}\| \leq 10^{-6}$ for three interaction strengths, three grid sizes and three time step sizes. These tables are used to describe, hereafter, the influences of interaction strength, grid size and time step size on the average number of coupling iterations per time step for the mono-fidelity coupling algorithms.

Table 7.1: Average number of coupling iterations per time step needed for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$ when using Gauss Seidel

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	12.00	6.42	4.39
	Medium	12.00	6.77	5.00
	Large	11.95	6.75	4.98
Medium	Small	12.85	6.47	4.35
	Medium	12.80	6.85	5.01
	Large	12.00	6.83	4.98
Strong	Small	16.15	6.95	4.91
	Medium	15.90	7.17	5.04
	Large	15.20	7.10	5.12

Table 7.2: Average number of coupling iterations per time step needed for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$ when using Aitken

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	12.00	6.32	4.37
	Medium	11.95	6.75	5.00
	Large	11.95	6.75	4.98
Medium	Small	12.60	6.35	4.31
	Medium	12.00	6.85	4.99
	Large	12.00	6.83	4.98
Strong	Small	15.10	6.70	4.56
	Medium	15.10	7.03	4.99
	Large	14.95	7.02	5.03

Table 7.3: Average number of coupling iterations per time step needed for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$ when using Broyden

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	9.70	6.13	4.34
	Medium	9.00	6.23	5.00
	Large	9.00	6.38	4.96
Medium	Small	10.00	6.13	4.31
	Medium	10.00	6.42	4.99
	Large	9.95	6.43	4.97
Strong	Small	11.95	6.50	4.73
	Medium	11.95	6.92	4.99
	Large	11.85	6.95	5.03

Table 7.4: Average number of coupling iterations per time step needed for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$ when using IQN-ILS(0)

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	9.00	6.10	4.34
	Medium	9.00	6.22	4.99
	Large	9.00	6.33	4.96
Medium	Small	9.05	6.10	4.29
	Medium	9.00	6.32	4.98
	Large	9.00	6.83	4.96
Strong	Small	11.00	6.33	4.54
	Medium	11.65	6.85	4.99
	Large	11.15	6.93	5.03

Influence of interaction strength on number of coupling iterations

The three levels of interaction strength are labeled "weak", "medium" and "strong" in the FSI column of Table 7.1 through Table 7.4. By comparing the averages reported for the weak and the medium interaction cases, it is observed that the medium cases, mostly, require more iterations to converge than the weak cases, however, the differences are quite small. The differences are more pronounced when comparing the averages for the strong cases, on the one hand, against the averages for the weak and medium cases, on the other hand. The strong cases require, mostly, more iterations to converge than weak and medium cases. This is especially true for the cases using the largest time step (i.e. $\Delta t = P_{cr}/10$). The influence of the interaction strength on the averages decreases as the size of the time step decreases. For the cases using the smallest time step (i.e. $\Delta t = P_{cr}/70$) the interaction strength has no patternized influence on the averages.

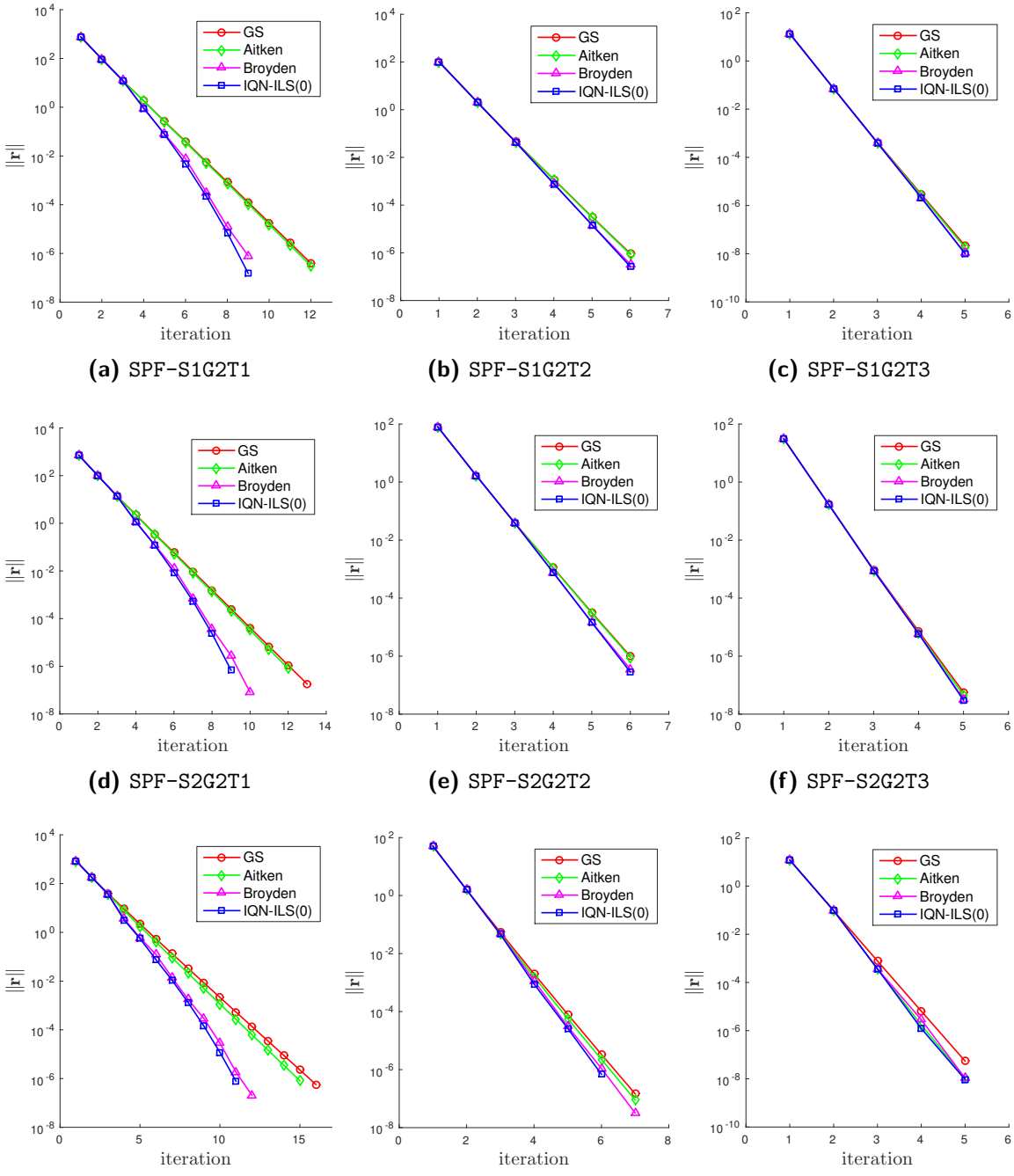
Influence of grid size on number of coupling iterations

On first sight, the grid sizes appear to have no significant influence on the average number of coupling iterations per time step. Some patterns can be discerned, however, they might be purely coincidental (i.e. product of the specific settings and/or problem used for the simulations). For the cases using the largest time step (i.e. $\Delta t = P_{cr}/10$), the averages remain either the same or decrease as the grid size increases. The only exception to this pattern, as seen in Table 7.4, is when IQN-ILS(0) is used for the strong interaction cases at the largest time step. The inverse pattern is followed by the quasi-Newton based algorithms (i.e. Broyden and IQN-ILS(0)) for $\Delta t = P_{cr}/30$; those averages, as observed from Table 7.3 and Table 7.4, increase as the grid size increases. The averages for the strong interaction cases using $\Delta t = P_{cr}/70$ increase as the grid size increases, for all mono-fidelity coupling algorithms.

Influence of time step on number of coupling iterations

The size of the time step has a significant influence on the average number of coupling iterations per time step. As the time step decreases, the averages decrease drastically. Also, as the size of the time step decreases the influences of the interaction strength and grid size become either less pronounced or insignificant.

Influence of coupling algorithm choice on coupling iterations



(a) SPF-S1G2T1 **(b)** SPF-S1G2T2 **(c)** SPF-S1G2T3

(d) SPF-S2G2T1 **(e)** SPF-S2G2T2 **(f)** SPF-S2G2T3

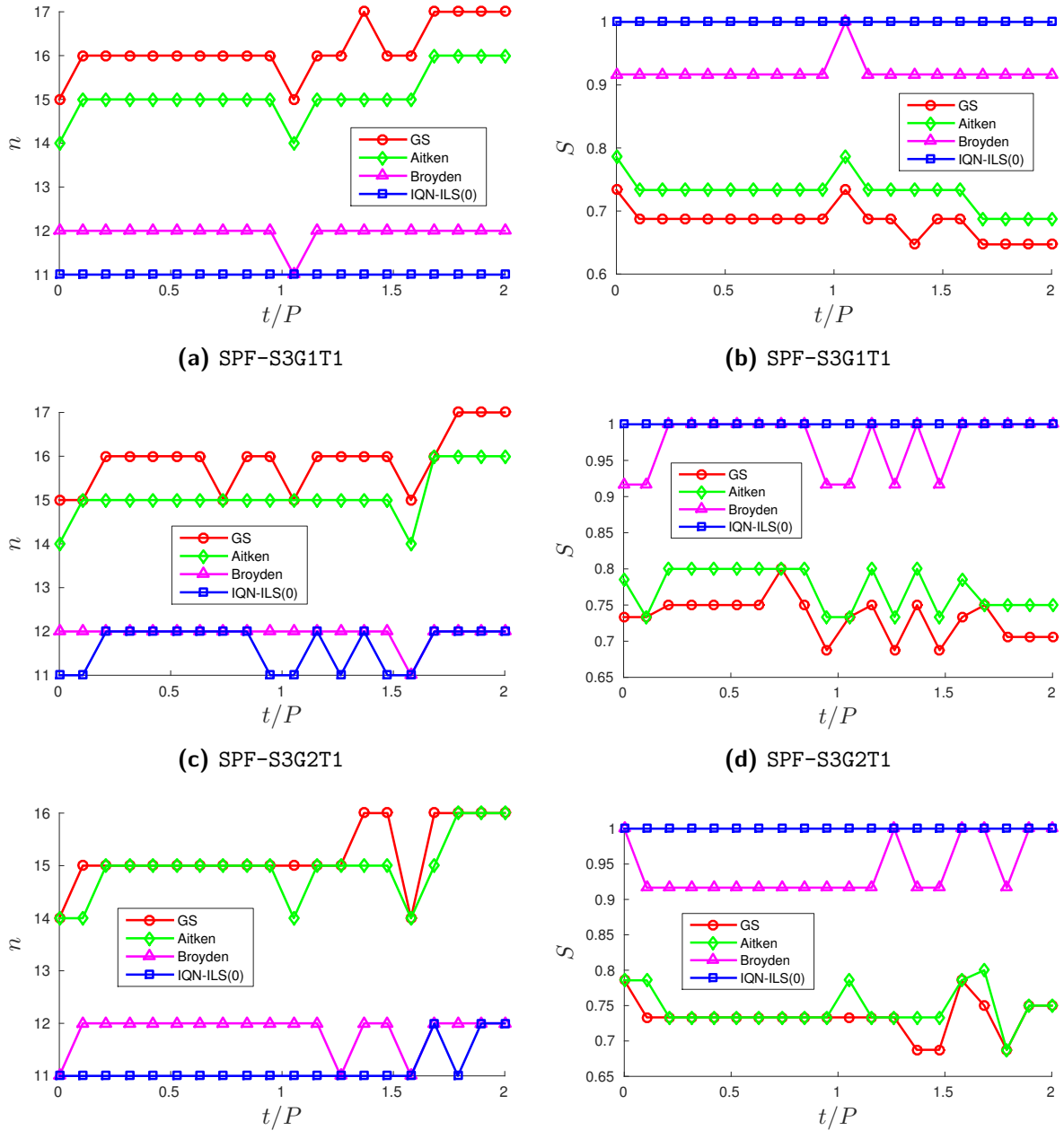
(g) SPF-S3G2T1 **(h)** SPF-S3G2T2 **(i)** SPF-S3G2T3

Figure 7.1: Residual histories for a representative time step of the supersonic panel flutter problem on a medium grid for several interactions strengths and several sizes for the time step

When comparing the results from Table 7.1 through Table 7.4 against one another, it is observed that IQN-ILS(0) is the most efficient, followed by Broyden, followed by Aitken and that Gauss Seidel is the least efficient. This is especially true for the cases using the largest time step (i.e. $\Delta t = P_{cr}/10$). The

averages of IQN-ILS(0) for the largest time step are the least effected by going from the weak to the medium interaction cases when compared to the other algorithms. For smaller time steps the overall differences become eiter less pronounced ($\Delta t = P_{cr}/30$) or insignificant ($\Delta t = P_{cr}/70$). Hence, for the cases using the smallest time step (i.e. $\Delta t = P_{cr}/70$) all mono-fidelity coupling algorithms perform about equally for the supersonic panel flutter problem.

The residual histories for all mono-fidelity coupling algorithms on a medium grid at a representative time are depicted in Figure 7.1 for all interaction strenghts and all time step sizes. The figures are consistent with the results reported in Table 7.1 through Table 7.4. The selected coupling algorithm becomes less important as the time step size decreases. The residual histories for all coupling algorithms in Figure 7.1c and Figure 7.1f almost entirely overlap. On the other hand, for the largest time step it is observed from Figure 7.1a, Figure 7.1d and Figure 7.1g that the residual norm decreases a lot quicker when using Broyden or IQN-ILS(0) instead of Gauss Seidel or Aitken. Furthermore, Gauss Seidel and Aitken converge linearly for all cases, Broyden and IQN-ILS(0) achieve superlinear convergence for the cases using the largest time step, as observed from Figure 7.1a, Figure 7.1d and Figure 7.1g. It is, however, noteworthy that the residuals reported by all the coupling algorithms are almost indistinguishable for the first three iterations. Differences become visible from the fourth iteration and become larger with further iterations. Another noteworthy observation is the resemblance of Broyden's slope to the slope of IQN-ILS(0) and the resemblance of Gauss Seidel's slope to the slope of Aitken. The quasi-Newton-based coupling algorithms (Broyden and IQN-ILS(0)) consistently outperform the fixed-point-iteration-based coupling algorithms (Gauss Seidel and Aitken).



(a) SPF-S3G1T1 **(b)** SPF-S3G1T1

(c) SPF-S3G2T1 **(d)** SPF-S3G2T1

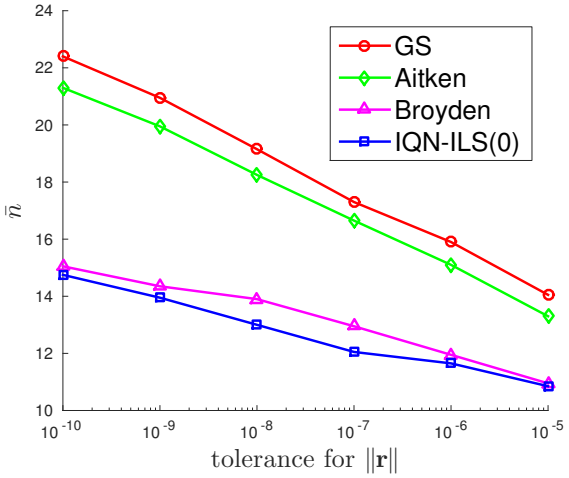
(e) SPF-S3G3T1 **(f)** SPF-S3G3T1

Figure 7.2: Performance comparison of mono-fidelity coupling algorithms relative to IQN-ILS(0), for SPF-S3G1T1, SPF-S3G2T1 and SPF-S3G3T1

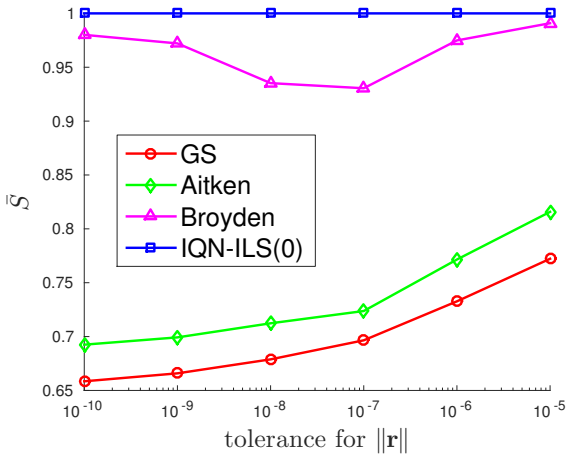
Figure 7.2 shows for the strong interaction cases using the largest time step the number of iterations needed for convergence up to $\|\mathbf{r}\| \leq 10^{-6}$ and the corresponding speeddown of the coupling algorithms relative to IQN-ILS(0) for all grid sizes. It is observed that the Gauss Seidel is mainly 26% - 31% less efficient than IQN-ILS(0). Aitken is about 20% - 27% less efficient than IQN-ILS(0). Broyden is almost as efficient as IQN-ILS(0), being 9% less efficient than IQN-ILS(0) at the worst case.

Influence of residual norm tolerance on number of coupling iterations

Figure 7.3 shows how the average number of coupling iterations per time step vary with residual norm tolerances. From Figure 7.3a it is observed that the increase of iterations for a decreasing tolerance is quite steep for the algorithms based on fixed-point iterations (Gauss Seidel and Aitken). The increase for the quasi-Newton-based algorithms is less steep. From Figure 7.3b one can see the relative efficiency of Gauss Seidel and Aitken decrease from 77% and 82% to 66% and 69%, respectively. The relative efficiency of Broyden drops only between $10^{-7} \leq \|\mathbf{r}\| \leq 10^{-5}$ and rises between $10^{-10} \leq \|\mathbf{r}\| \leq 10^{-7}$. Broyden remains quite efficient compared to IQN-ILS(0) while Gauss Seidel and Aitken become very inefficient for small tolerances.



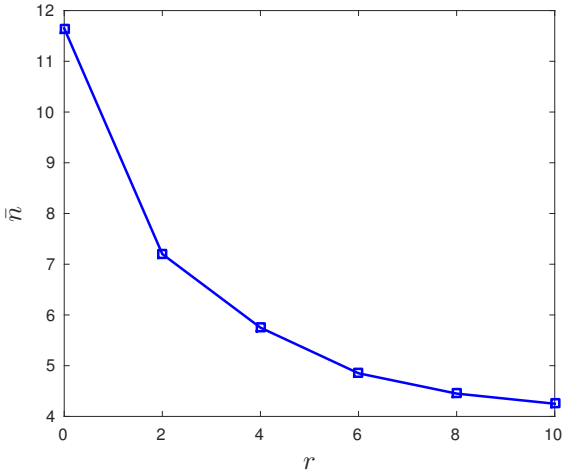
(a) Average number of iterations per time step for various tolerances



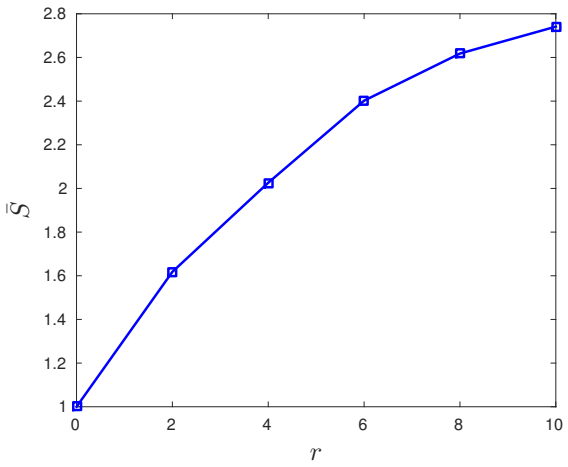
(b) Efficiency of coupling algorithms relative to IQN-ILS(0) for various tolerances

Figure 7.3: Comparison of the number of coupling iterations needed by mono-fidelity coupling algorithms to converge `SPF-S3G2T1` for various tolerances

Influence of r in IQN-ILS(r) on number of coupling iterations



(a) Average number of coupling iterations per time step needed by IQN-ILS(r) for various number of reuse, r



(b) Speedup of IQN-ILS(r) relative to IQN-ILS(0) for various number of reuse, r

Figure 7.4: Performance comparison of IQN-ILS(r) relative to IQN-ILS(0) for `SPF-S3G2T1`

The performance of IQN-ILS(0) can be improved by reusing data from previous iterations. The average number of coupling iterations per time step it takes IQN-ILS(r) to converge `SPF-S3G2T1` is depicted in Figure 7.4a for various number of reuse, r. The speedup of IQN-ILS(r) relative to IQN-ILS(0) is shown in Figure 7.4b. From these figures it is observed that the reuse of data from previous time steps results in a large improvement over IQN-ILS(0). Reusing data from four previous time steps already cuts the number of coupling iterations in half. Reusing data from ten previous time steps results in a speedup north of 2.7. The speedup as function of reuse, as seen in Figure 7.4b, is sublinear as expected. This is due to the fact that converging after a single coupling iteration is the absolute minimum that can be achieved.

Discussion of mono-fidelity results

The results shown for mono-fidelity simulations of the supersonic panel flutter problem clearly indicate that IQN-ILS(0) outperforms Gauss Seidel, Aitken and Broyden when large time steps are used. The use of data from all previous iterations of the current time step gives IQN-ILS(0) its great performance characteristics. The superiority of IQN-ILS(0) over the other algorithms becomes less pronounced with decreasing time step sizes as expected for a a compressible flow. When using very small time steps, the displacement of the structure in between two time levels is relatively small, hence, the corresponding flow perturbation and feedback to the structure will be small as well. Thus, for sufficiently small time steps it is as if a weak interaction occurs between two time levels. Adding data from previous iterations to improve the inverse-Jacobian approximation then becomes superfluous. A simple algorithm like Gaus Seidel will perform similar to more advanced algorithms like IQN-ILS(0) in such cases. The opposite is, however, true for incompressible flows where pertubations occur instantaneously instead of travelling along a wave with a finite advection velocity. For incompressible flows the FSI strength tends to increase with decreasing time step sizes.

Grid sizes appear to have no significant influence on the performance of the mono-fidelity coupling algorithms when simulating the FSI of the supersonic panel flutter problem. This result is simply understood from the absence of an interpolation method to transfer interface quantities between the

fluid and the structure mesh. Interpolation is not required because the fluid and the structure mesh match at the interface. Thus, there is no spatial discrepancy between interface quantities defined on the fluid- or structure-side of the interface. The coupling of the supersonic flow with the flexible panel is, hence, purely a temporal problem.

Increasing the FSI strength has the expected effect on the average number of coupling iterations per time. All mono-fidelity coupling algorithms required more iterations to converge as the FSI strength is increased. The increase, however, appears to be steeper for the coupling algorithms based on fixed-point iterations like Gauss Seidel and Aitken than for Broyden and IQN-ILS(0). This result agrees with literature reporting on superlinear convergence properties of IQN-ILS(0). Because Broyden is based on Newton's method, it is natural to expect Broyden to perform more similarly to IQN-ILS(0) than to Gauss Seidel and Aitken. The results reported in this chapter for Broyden verify that expectation.

Decreasing the residual norm tolerance has the same effect on the performance of the coupling algorithms as increasing the FSI strength. The average number of coupling iterations per time step increase and the increase is more severe for Gauss Seidel and Aitken than for Broyden and IQN-ILS(0). The superior performance of Broyden and IQN-ILS(0) over Gauss Seidel and Aitken for stricter tolerances is due to their ability to achieve superlinear convergence.

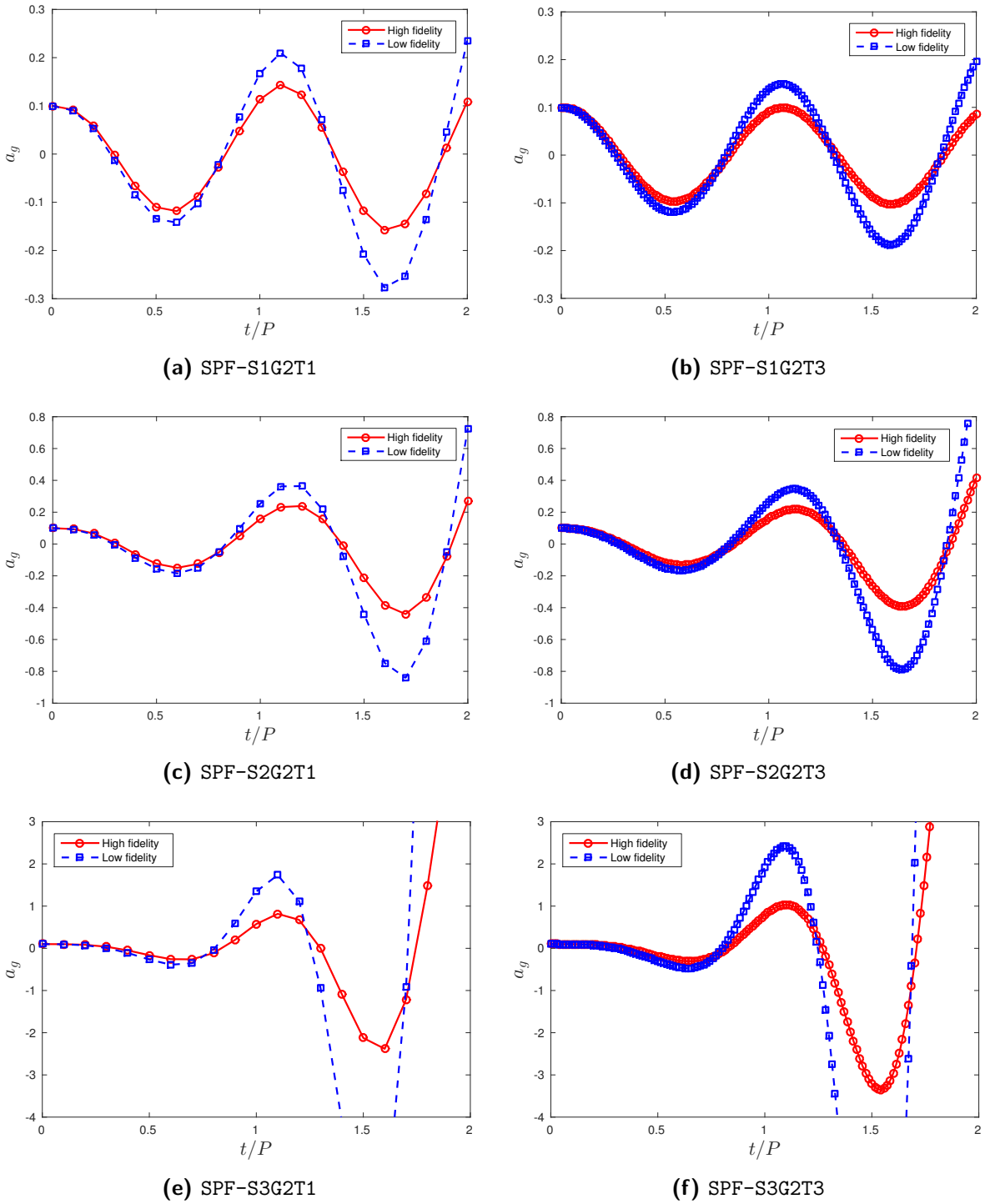
Under-relaxation (i.e. $0 < \omega < 1$) is used to make fixed-point iterations more stable. Due to a dynamic under-relaxation factor, Aitken is expected to be more stable and to converge faster than Gauss Seidel which does not incorporate any relaxation (i.e. $\omega = 1$). Although Aitken did not perform worse than Gauss Seidel in any case, the performance of Aitken for the supersonic panel flutter problem is only marginally better than that of Gauss Seidel. The reason for this is revealed by inspecting the value for the under-relaxation factor, which was found to be in the range of $0.88 - 0.97$. Hence, the optimal value for the under-relaxation factor for the supersonic panel flutter problem lies close to the constant that Gauss Seidel is using. This explains the small difference in performance between Gauss Seidel and Aitken for the supersonic panel flutter problem.

The slightly better performance of IQN-ILS(0) relative to Broyden is explained by the former having a larger pool of data to use for the approximation of the inverse-Jacobian. The performance of IQN-ILS is greatly enhanced by reusing data from previous time steps. As IQN-ILS(0) benefits from a larger pool of data when compared to Broyden, IQN-ILS(r) benefits from an even larger pool of data. The sublinear increase of speedup as function of the number of reuse is expected because the number of coupling iterations needed for convergence cannot be lower than one.

Reusing data from previous time steps indefinetely is not wise for mainly two reasons. Firstly, if the dynamics of the system changes drastically over time, the relevance of data from a previous time step diminishes with each subsequent time step. Including data from too many time steps back might therefore deteriorate the performance of IQN-ILS(r). Secondly, from an algorithmic point of view, the QR-decomposition is at the heart of IQN-ILS. The QR-decomposition scales with $2mn^2$ flops, where m is the number of rows and n is the number of colomns. The number of rows is mainly dictated by the number of nodes on the interface and is, hence, fixed throughout the entire simulation. The number of columns equals the number of coupling iterations performed in the current time step plus the number of coupling iterations performed in each previous time step added to the data pool. As the number of flops required IQN-ILS scales quadratically with the number of coupling iterations, the corresponding CPU time might become non-negligible compared to the CPU time needed to evaluate the interface residual. This violates the assumption that computing a quasi-Newton update is cheap compared to evaluating the external black-box solvers. In such a case, the CPU time spend on computing a quasi-Newton update should be incorparated into the efficiency analysis of different coupling algorithms.

7.1.2 Multi-fidelity results

Comparison of high- and low-fidelity panel responses



(a) SPF-S1G2T1 **(b)** SPF-S1G2T3

(c) SPF-S2G2T1 **(d)** SPF-S2G2T3

(e) SPF-S3G2T1 **(f)** SPF-S3G2T3

Figure 7.5: Visualisation of the panel behaviour on a medium grid, for several interaction strengths and time step sizes

Before exploiting a low-fidelity model to accelerate the coupling procedure of a high-fidelity model it is important to verify whether the low-fidelity model behaves similar to the high-fidelity model. If the behaviours are completely different it will be unlikely for spacemapping to provide any speedup relative to a mono-fidelity simulation.

To examine the panel behaviour of the low- and high-fidelity models for the supersonic panel flutter problem a generalised coordinate is introduced

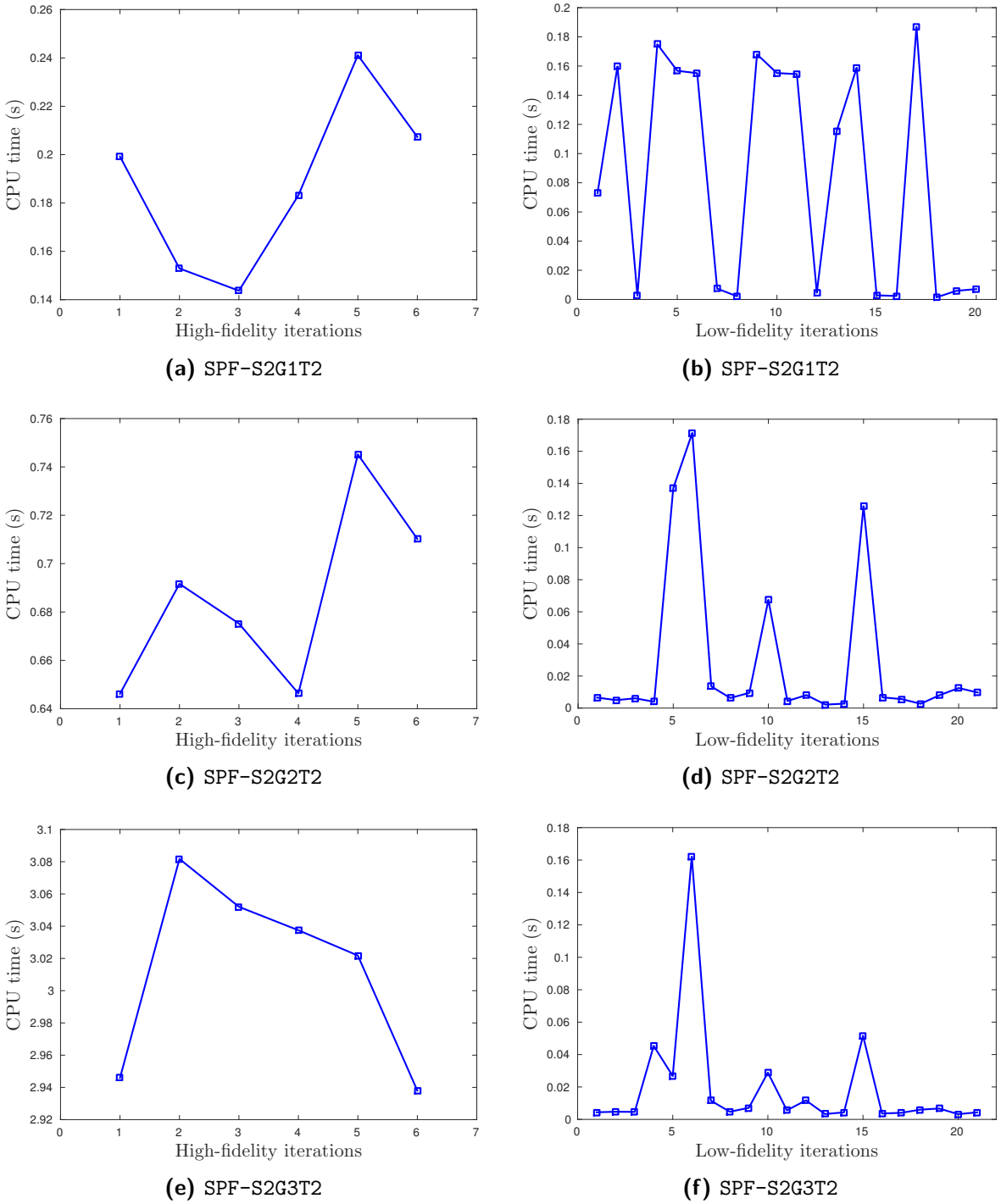
$$a_g(t) = \xi^T w(t), \tag{7.1}$$

where a_g is the projection of the panel displacement, w, on the eigenmode, ξ, shown in Figure 6.3.

The panel behaviour for both models are shown for all interaction strengths and the the largest and smallest time step sizes in Figure 7.5. The general behaviour of both models agrees well for all interaction strenghts. Flutter is observed for both models for all interaction strenghts, except for the high-fidelity model when using the smallest time step for the weak interaction as seen in Figure 7.5b. The low-fidelity model is consistenly less damped than the high-fidelity model. This results in larger oscillations for the low-fidelity response. By comparing Figure 7.5a to Figure 7.5b and Figure 7.5c to Figure 7.5d it can be seen for the weak and medium interaction strengths, respectively, that both models are more damped when using smaller time steps. The converse is true for the strong interaction as observed by comparing Figure 7.5e to Figure 7.5f. The difference in amplitudes obtained by the models starts out insignificantly small and grows with time. The growth of the difference in amplitudes is most pronounced for the strong interaction.

Although the response of the low-fidelity model deviates from the high-fidelity response, the general behaviour of both models is similar. Except for the single exception, both models predict flutter at roughly the same critical frequency.

Comparison of CPU time for high- and low-fidelity residual operator evaluation



(a) SPF-S2G1T2 **(b)** SPF-S2G1T2

(c) SPF-S2G2T2 **(d)** SPF-S2G2T2

(e) SPF-S2G3T2 **(f)** SPF-S2G3T2

Figure 7.6: CPU times for the evaluation of the high-fidelity and the low-fidelity residual operator on all three grid sizes

An easy way to measure the speedup obtained by accelerating a coupling algorithm using Aggressive Space Mapping is to compare CPU times needed for the evaluations of the low- and the high-fidelity

residual operator. However, for academic test cases, such as the supersonic panel flutter, the evaluation of the high-fidelity residual is not very computational expensive. As a consequence, the CPU times recorded for the, even cheaper, evaluations of the low-fidelity residual operator can be polluted by other processes running on the computer (e.g. output to screen, logging, read/write from/to harddisk, system services).

In Figure 7.6 the recorded CPU times for the evaluation of the high-fidelity and low-fidelity residual operators are shown for a single representative time step on all three grid sizes. It is observed that, especially, for the low-fidelity iterations that there is a large difference between the smallest and highest recorded CPU time on all three grids (see Figure 7.6b, Figure 7.6d and Figure 7.6f). As the grid size increases the number of low-fidelity iterations for which a relatively high CPU time is reported decreases.

Table 7.5: Ratio of the maximum to the minimum recorded CPU time needed for a residual operator evaluation at a representative time step

	Small grid	Medium grid	Large grid
High-fidelity	1.68	1.15	1.05
Low-fidelity	131.32	79.37	50.07

The ratio of the highest to the lowest CPU time for both models on all three grids are collected in Table 7.5. The ratio decreases for both models as the grid size increases. However, even at the largest grid size the ratio is still very high for the low-fidelity iterations. This implies that even the largest grid is too small for an accurate speedup estimation based on CPU times. As the ratio becomes more favourable (i.e. decreasing towards 1) as the grid size increases, it is more representative for an industrial application to base the speedup on the number of high-fidelity iterations instead of comparing CPU times. Therefore, speedups reported in this section for the supersonic panel flutter problem are defined as

$$S = \frac{\#\text{ high-fidelity iterations by [ASM-QN]}}{\#\text{ high-fidelity iterations by [QN]}}, \tag{7.2}$$

where [ASM-QN] is a placeholder for the ASM-accelerated quasi-Newton coupling algorithm (i.e. multi-fidelity coupling algorithm) being investigated and [QN] is a placeholder for the reference coupling algorithm which is not ASM-accelerated (i.e. mono-fidelity coupling algorithm).

Comparison of multi-fidelity speedup relative to mono-fidelity counterpart

Table 7.6 reports the speedup of ASM-GS relative to Gauss Seidel. ASM-GS uses Gauss Seidel for both the outer and the inner iterations.

A speedup is achieved only for the cases where the largest and the medium time steps are used. Speedups ranging between 1.40-1.61 are achieved when using the largest time step. For the medium time step the speedup is at best 1.15 and in the worst case (i.e. `SPF-S3G1T2`) there is a speeddown of 0.99. When the smallest time step is used no speedup is achieved at all, speeddowns down to 0.88 are observed. Decreasing the size of the time step, hence, seems to have a negative effect on the speedup for ASM-GS.

Grid sizes and interaction strenghts appear to have no patternised and/or significant effect on the speedup of ASM-GS relative to Gauss Seidel.

Table 7.6: Speedup of ASM-GS relative to Gauss-Seidel for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	1.47	1.07	0.99
	Medium	1.51	1.14	1.00
	Large	1.52	1.14	1.00
Medium	Small	1.61	1.07	0.99
	Medium	1.60	1.15	1.00
	Large	1.41	1.15	0.99
Strong	Small	1.40	0.99	0.90
	Medium	1.56	1.03	0.93
	Large	1.40	1.01	0.88

Table 7.7 reports the speedup of ASM-Aitken relative to Aitken. ASM-Aitken uses Aitken for the outer iterations and Gauss Seidel for the inner iterations. Speedups are achieved by ASM-Aitken for all interaction strenghts on all grids and for all time step sizes, except for `SPF-S3G1T3`.

Large speedups in the range of 1.88-2.09 are achieved by ASM-Aitken when using the largest time step. This implies, for the largest time step, that accelerating Aitken with Aggressive Space Mapping cuts the number of high-fidelity iterations in half. Decent speedups in the range of 1.14-1.37 are obtained by ASM-Aitken for the cases using the medium time step. Speedups are also observed in the cases for which the smallest time step were used, except for `SPF-S3G1T3`. Speedups achieved by ASM-Aitken for the smallest time step are in the range of 1.07-1.22. Hence, also for ASM-Aitken a decrease in the size of the time step has a negative effect on the speedup. However, even for the smallest time step decent speedups are achieved.

Grid size appear to have no consistent effect on the speedup achieved by ASM-Aitken. The largest differences caused by grid sizes are observed between the `SPF-S2G1T1`, `SPF-S2G2T1` and `SPF-S2G3T1` cases where the speedups are 1.88, 1.74 and 2.00 respectively.

The difference in speedups in between the weak interaction cases and the medium interaction cases are insignificant except for those which use the largest time step. The differences in between the medium and the strong interaction cases are, however, noticeable. Speedups for the strong interaction cases are 1.12-1.16 lower than for the medium interaction cases. The exception is again those few cases using the largest time step, where the speedups for the strong interaction case are acutally higher.

Table 7.7: Speedup of ASM-Aitken relative to Aitken for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	2.00	1.26	1.07
	Medium	1.99	1.35	1.21
	Large	1.99	1.36	1.22
Medium	Small	1.88	1.26	1.06
	Medium	1.74	1.37	1.20
	Large	2.00	1.37	1.17
Strong	Small	1.91	1.14	0.96
	Medium	2.07	1.21	1.10
	Large	2.09	1.22	1.07

Table 7.8 reports the speedup of ASM-Broyden relative to Broyden. ASM-Broyden uses Broyden for

the outer iterations and Gauss Seidel for the inner iterations. Speedups are achieved by ASM-Broyden for all interaction strenghts on all grids and for all time step sizes, except for `SPF-S3G1T3` `SPF-S3G2T3`.

Speedups achieved by ASM-Broyden for the largest time step are mostly in the range of 1.40-1.50. Speedups for the medium time step are mostly in the range of 1.21-1.29, except for the strong interaction cases. The smallest speedups are achieved for the smallest time step. Speedups for the smallest time step are mostly negligible except for `SPF-S2G1T3` and `SPF-S2G2T3` for which a speedup of 1.18 and 1.17 are achieved, respectively.

Grid sizes and interaction strengths appear to have no significant consistent effect on the speedups achieved by ASM-Broyden.

Table 7.8: Speedup of ASM-Broyden relative to Broyden for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	1.41	1.23	1.03
	Medium	1.50	1.25	1.18
	Large	1.50	1.29	1.10
Medium	Small	1.43	1.21	1.02
	Medium	1.44	1.27	1.17
	Large	1.53	1.29	1.07
Strong	Small	1.46	1.09	0.96
	Medium	1.38	1.19	1.00
	Large	1.49	1.18	1.02

Table 7.9 reports the speedup of ASM-ILS(0) relative to IQN-ILS(0). ASM-ILS(0) uses IQN-ILS(0) for the outer iterations and Gauss Seidel for the inner iterations. Even though ASM-ILS(0) achieves a speedup for all cases, the speedups are mostly negligible with exception of `SPF-S1G2T1`, `SPF-S1G3T1`, `SPF-S3G2T2` and `SPF-S3G2T3` for which a speedup of 1.12, 1.13, 1.11 and 1.10 are achieved, respectively.

Table 7.9: Speedup of ASM-ILS(0) relative to IQN-ILS(0) for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	1.01	1.05	1.05
	Medium	1.12	1.05	1.09
	Large	1.13	1.07	1.03
Medium	Small	1.01	1.04	1.04
	Medium	1.01	1.06	1.08
	Large	1.02	1.08	1.02
Strong	Small	1.00	1.06	1.04
	Medium	1.07	1.11	1.06
	Large	1.05	1.10	1.03

Comparison of multi-fidelity speedup relative to IQN-ILS(0)

To compare the performance of the different variants for Aggressive Space Mapping the speedups reported in Table 7.10, Table 7.11, Table 7.12 and Table 7.9 for, respectively, ASM-GS, ASM-Aitken, ASM-Broyden and ASM-ILS(0) are computed relative to the number of high-fidelity iterations needed

by IQN-ILS(0) as reported in Table 7.4.

When comparing ASM-GS to IQN-ILS(0) the former is mostly less efficient when using the smallest time step, mostly very slightly more efficient when using the medium time step and mostly somewhat more efficient when using the largest time step. For a few cases ASM-GS achieves a speedup north of 1.10 when compared to IQN-ILS(0).

ASM-Aitken is more efficient than IQN-ILS(0) for all cases except for `SPF-S3G1T3`. Large speedups of 1.35-1.60 are achieved by ASM-Aitken relative to IQN-ILS(0) when using the largest time step. Even when using the smallest time step speedups of about 1.20 are observed for ASM-Aitken relative to IQN-ILS(0).

Speedups achieved by ASM-Broyden are similar to those of ASM-Aitken. Suprisingly, ASM-Aitken is mostly more efficient than ASM-Broyden, although, the differences are mostly not considerable.

ASM-ILS(0) is unique in achieving speedups relative to IQN-ILS(0) for all cases without exception. Unfortunately, the performance of ASM-ILS(0) is either inferior to ASM-GS in the worst case or roughly the same in the best case.

Table 7.10: Speedup of ASM-GS relative to IQN-ILS(0) for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$

FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	1.10	1.02	0.98
	Medium	1.13	1.05	1.00
	Large	1.50	1.07	1.00
Medium	Small	1.13	1.01	0.97
	Medium	1.13	1.06	0.99
	Large	1.06	1.07	0.99
Strong	Small	0.96	0.90	0.83
	Medium	1.14	0.98	0.92
	Large	1.03	0.98	0.87

Table 7.11: Speedup of ASM-Aitken relative to IQN-ILS(0) for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$

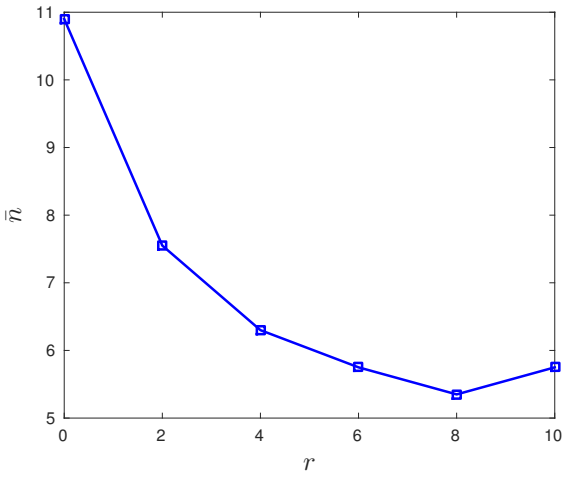
FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	1.50	1.22	1.07
	Medium	1.50	1.24	1.21
	Large	1.50	1.28	1.22
Medium	Small	1.35	1.21	1.06
	Medium	1.30	1.26	1.19
	Large	1.50	1.28	1.17
Strong	Small	1.39	1.07	0.96
	Medium	1.60	1.18	1.10
	Large	1.56	1.21	1.07

Table 7.12: Speedup of ASM-Broyden relative to IQN-ILS(0) for the convergence of the supersonic panel flutter problem up to $\|\mathbf{r}\| \leq 10^{-6}$

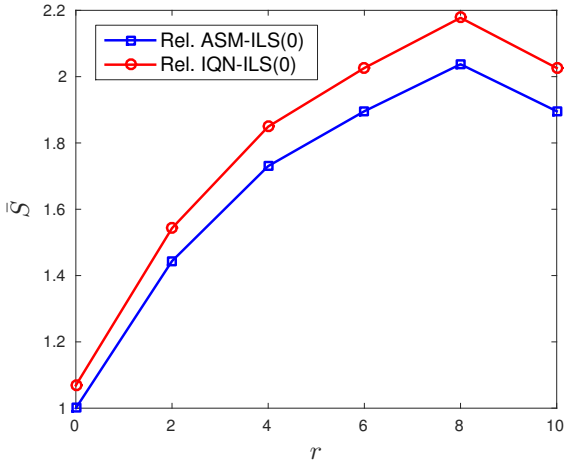
FSI	Grid	$\Delta t = P_{cr}/10$	$\Delta t = P_{cr}/30$	$\Delta t = P_{cr}/70$
Weak	Small	1.30	1.23	1.03
	Medium	1.50	1.25	1.18
	Large	1.50	1.29	1.10
Medium	Small	1.29	1.20	1.01
	Medium	1.30	1.26	1.17
	Large	1.38	1.28	1.07
Strong	Small	1.34	1.06	0.92
	Medium	1.35	1.17	1.00
	Large	1.40	1.18	1.02

Comparison of ASM-ILS(r) relative to IQN-ILS(0) and ASM-ILS(0)

The average number of high-fidelity iterations needed by ASM-ILS(r) for various number of reuse, r, to converge `SPF-S3G2T1` up to a residual norm of $\|\mathbf{r}\| \leq 10^{-6}$ is depicted in Figure 7.7a. The speedup of ASM-ILS(r) relative to IQN-ILS(0) and relative to ASM-ILS(0) are shown in Figure 7.7b. Speedups are observed up to nearly 2.20 (relative to IQN-ILS(0)) when reusing data from 8 previous time steps. Increasing the number of reuse beyond 8 decreases the speedup slightly, in this case, to slightly north of 2.00 (relative to IQN-ILS(0)). In this case, the speedup reported for $r = 6$ is about the same to the speedup reported for $r = 10$.



(a) Average number of high-fidelity iterations per time step needed by ASM-ILS(r) for various number of reuse, r



(b) Speedup of ASM-ILS(r) relative to IQN-ILS(0) and ASM-ILS(0) for various number of reuse, r

Figure 7.7: Performance comparison of ASM-ILS(r) relative to IQN-ILS(0) and ASM-ILS(0) for `SPF-S3G2T1`

Minimising number of inner iterations per outer iteration

Little attention is paid to the inner iterations when the CPU time needed for evaluating the low-fidelity residual operator is very small compared the CPU time needed for evaluating the high-fidelity residual

operator. However, if the number of inner iterations per outer iteration is considerably high, the time spend on the low-fidelity model can become non-negligible. This will adversely effect the speedup of a multi-fidelity coupling algorithm.

A straightforward way to ensure a low number of inner iterations per outer iteration is to provide a good initial guess for the inner iterations. Using the initial guess for the outer iterations also as an initial guess for the inner iterations is a poor choice if the solution of the inner problem does not closely resemble the solution of the outer problem. Furthermore, the outer iterations try to solve

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in X} \| \mathcal{R}(\mathbf{x}) \|, \tag{7.3}$$

whereas the inner iterations try to solve

$$\mathbf{z} = \arg\min_{\mathbf{z}\in Z} \| \mathcal{R}(\mathbf{x}) - \tilde{\mathcal{R}}(\mathbf{z}) \| . \tag{7.4}$$

In words, the outer iterations try to find the high-fidelity solution for which the high-fidelity residual norm is minimal, whereas the inner iterations try to find the low-fidelity solution for which the low-fidelity residual resembles the most recent high-fidelity residual.

A good initial guess for the inner iterations can be derived by comparing the Gauss Seidel update

$$\mathbf{x}_{GS}^{i+1} = \mathbf{x}_{GS}^{i} + \mathbf{r}^i, \tag{7.5}$$

to the ASM-accelerated Gauss Seidel update

$$\mathbf{x}_{ASM}^{i+1} = \mathbf{x}_{ASM}^{i} + (\mathbf{z}^* - \mathbf{z}^i). \tag{7.6}$$

When the high-fidelity iterate, $\mathbf{x}^{i+1}$ is near the converged solution $\mathbf{x}^*$ one can assume

$$\mathbf{x}_{GS}^{i+1} - \mathbf{x}_{GS}^{i} \approx \mathbf{x}_{ASM}^{i+1} - \mathbf{x}_{ASM}^{i} \tag{7.7}$$

which implies

$$\mathbf{r}^i \approx \mathbf{z}^* - \mathbf{z}^i \rightarrow \mathbf{z}^i \approx \mathbf{z}^* - \mathbf{r}^i. \tag{7.8}$$

Using $\mathbf{z} = \mathbf{z}^* - \mathbf{r}^i$ as initial guess for the inner iterations, should provide an initial guess that improves for each subsequent outer iteration of a single time step. This behaviour is shown in Figure 7.8, where the number of inner iterations per outer iteration of a representative time steps is shown.

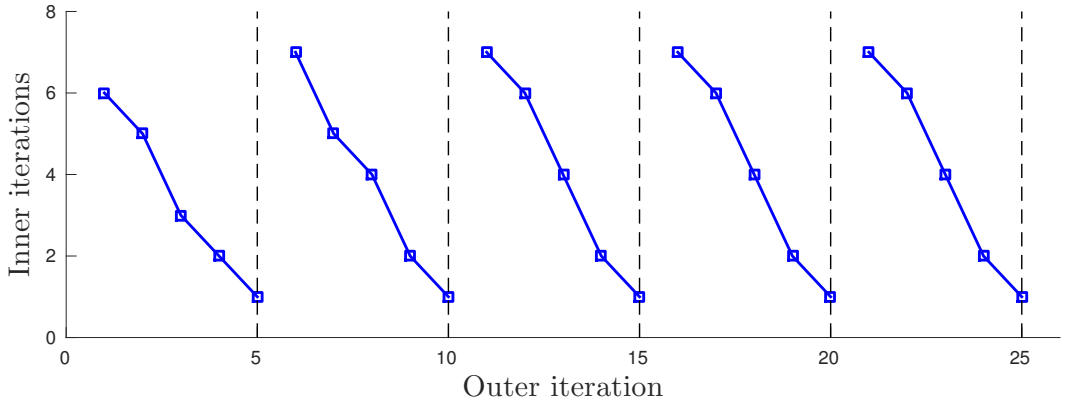


Figure 7.8: Decrease of number of inner iterations per outer iteration for 5 representative time steps when using the initial guess $\mathbf{z} = \mathbf{z}^* - \mathbf{r}^i$ in ASM-Aitken for `SPF-S2G2T2`

Note that the high-fidelity residual, $\mathbf{r}^i$, should be restricted if the low-fidelity model has less nodes on the fluid-structure interface than the high-fidelity model. Due to the computational work associated with interpolation, the proposed method might not be a viable option in such a case.

Another choice for an initial guess would be the solution of the inner problem from the previous outer iteration. This option is, however, not available for the first outer iteration and should therefore be accompanied by another type of initial guess for the first outer iteration.

Further investigation is required to determine which option performs better. Such an investigation is, however, outside of the scope of this project.

Discussion of multi-fidelity results

The panel response obtained from the high- and low-fidelity models for the supersonic panel flutter problem agree well. Although, the amplitude of the flutter behaviour is less damped for the low-fidelity model, the general behaviour is similar for the high-fidelity model. The piston response model is therefore a sufficiently accurate low-fidelity flow model for the supersonic panel flutter problem at the Mach numbers used for the simulations (i.e. 2.27, 2.28 and 2.33).

The computational power needed for the simulation of the supersonic panel flutter problem is quite low due to the relatively low number of fluid grid points and structure nodes. CPU timings of the evaluation of both the high- and the low-fidelity residual operator are therefore polluted by other (background) processes running on the computer. The CPU timings for evaluation of the low-fidelity residual operator are effected most severly. This renders CPU time as an inaccurate measure to be used for the determination of the speedup achieved with Aggressive Space Mapping. Increasing the grid size has a positive effect on the accuracy of the CPU timings. However, even when using the largest grid, the ratio of the longest CPU time to the lowest CPU time for the evaluation of the low-fidelity residual operator is about a factor 50.

Basing the speedup on the reduction of high-fidelity coupling iterations, speedups are achieved by all ASM-accelerated coupling algorithms. The best results are obtained by ASM-Aitken, followed by ASM-Broyden, followed by a third place shared by ASM-GS and ASM-ILS(0). ASM-Aitken and AMS-Broyden achieve large speedups of 1.50 relative to IQN-ILS(0) when using the largest time step. Even for the smallest time steps speedups north of 1.20 are achieved by ASM-Aitken and AMS-Broyden relative to IQN-ILS(0).

The unexpected poor performance of ASM-ILS(0) comes at a shock and cannot be currently clarified. An implementation error is suspected to be the culprit. Fortunately, the reuse of data from previous time steps still provides a decent speedup improvement up to 2.00 when reusing 8 previous time steps. However, unaccelerated reuse, currently, trumps ASM-accelerated reuse. Suspicion is that the same aforementioned implementation error is the cause of this.

A simple inexpensive and, most importantly, good initial guess for the inner iterations was proposed. The initial guess is derived from comparing the Gauss Seidel update to the ASM-accelerated Gauss Seidel update. The initial guess improves during a time step for each subsequent outer iteration. As a consequence, the number of inner iterations per outer iteration decreases for each subsequent outer iteration in a single time step. This result may appear rather insignificant, at first, when one advocates that the inner iterations should be a lot cheaper than the outer iterations. On the one hand, there may be cases where that is not uphold and on the other hand, if the number of inner iterations becomes excessive it might still adversely effect the speedup. Reducing the number of inner iterations makes more accurate (however, also more expensive) low-fidelity models viable. As the difference between the high- and the low-fidelity models decreases, the number of outer iterations should decrease as well. Even if the additional speedup is not large, the improvement is obtained by merely a single vector addition per outer iteration. Hence, the improvement is worth the considerably low effort.

7.2 Quasi-one-dimensional channel flow

Results obtained from simulating the channel flow problem using mono-fidelity coupling algorithms are presented in Section 7.2.1. Multi-fidelity results are presented in Section 7.2.2. Multi-fidelity results include the application of several algorithms for the outer iterations. For the inner iterations Aitken is used exclusively.

7.2.1 Mono-fidelity results

Physical results

The distributions of the radial displacement, fluid pressure and axial flow velocity of the quasi-one-dimensional channel forced by a periodic inflow at the left boundary are shown for six representative time levels in Figure 7.9. The inflow acts as a sinusoidal velocity wave that convects from left to right. The increase in axial velocity (and mass flow) causes the pressure acting on the inside of the channel wall to increase which causes the channel to deform radially outward. When the flow velocity decreases the pressure and radial deformation decrease accordingly.

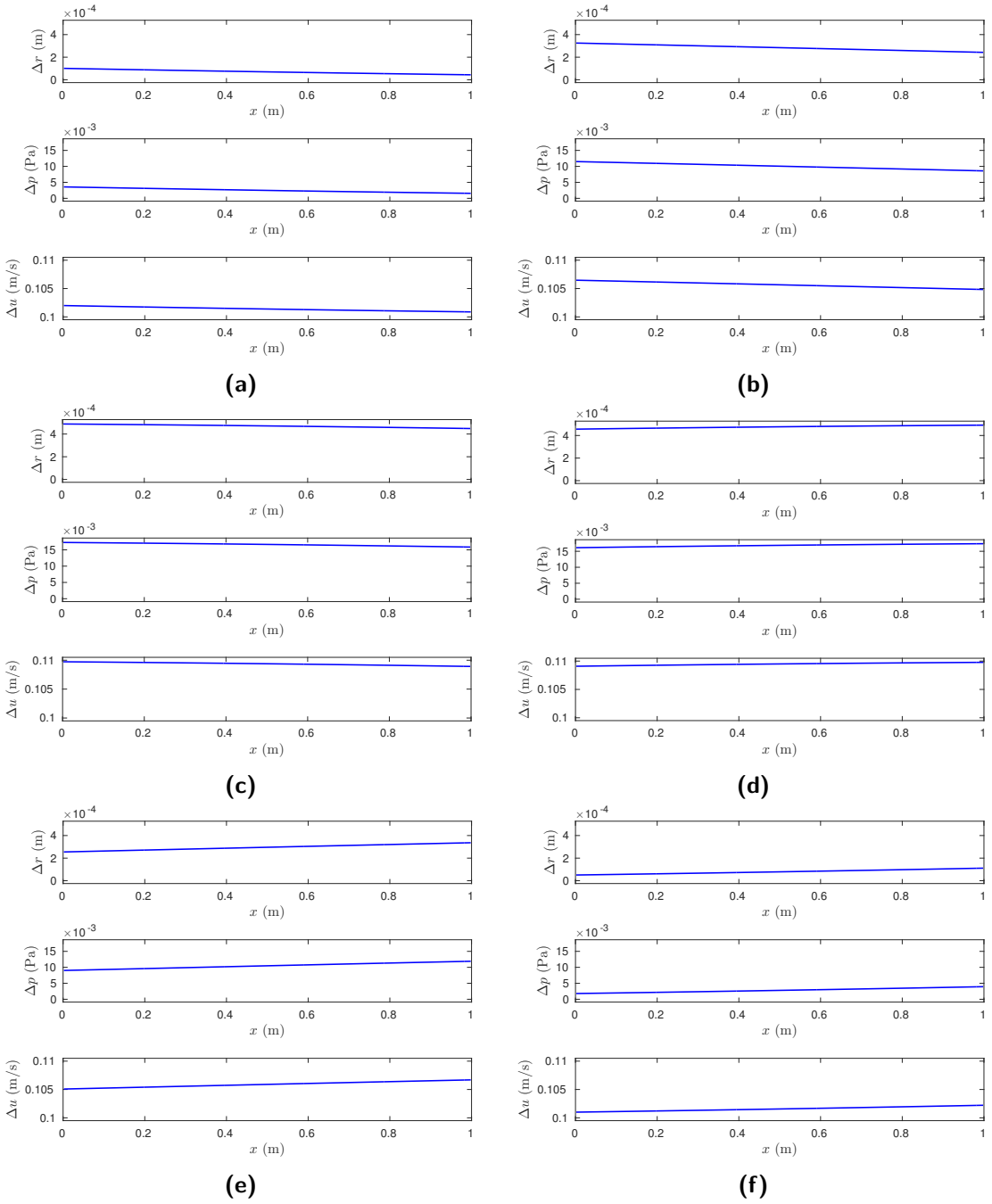


Figure 7.9: Distributions of the flow velocity, fluid pressure and radial structure displacement at (a) $\bar{t} = 0.15$, (b) $\bar{t} = 0.30$, (c) $\bar{t} = 0.45$, (d) $\bar{t} = 0.60$, (e) $\bar{t} = 0.75$, (f) $\bar{t} = 0.90$, where $\bar{t}$ is a fraction of the period of the periodic inflow.

Influence of coupling algorithm choice on coupling iterations

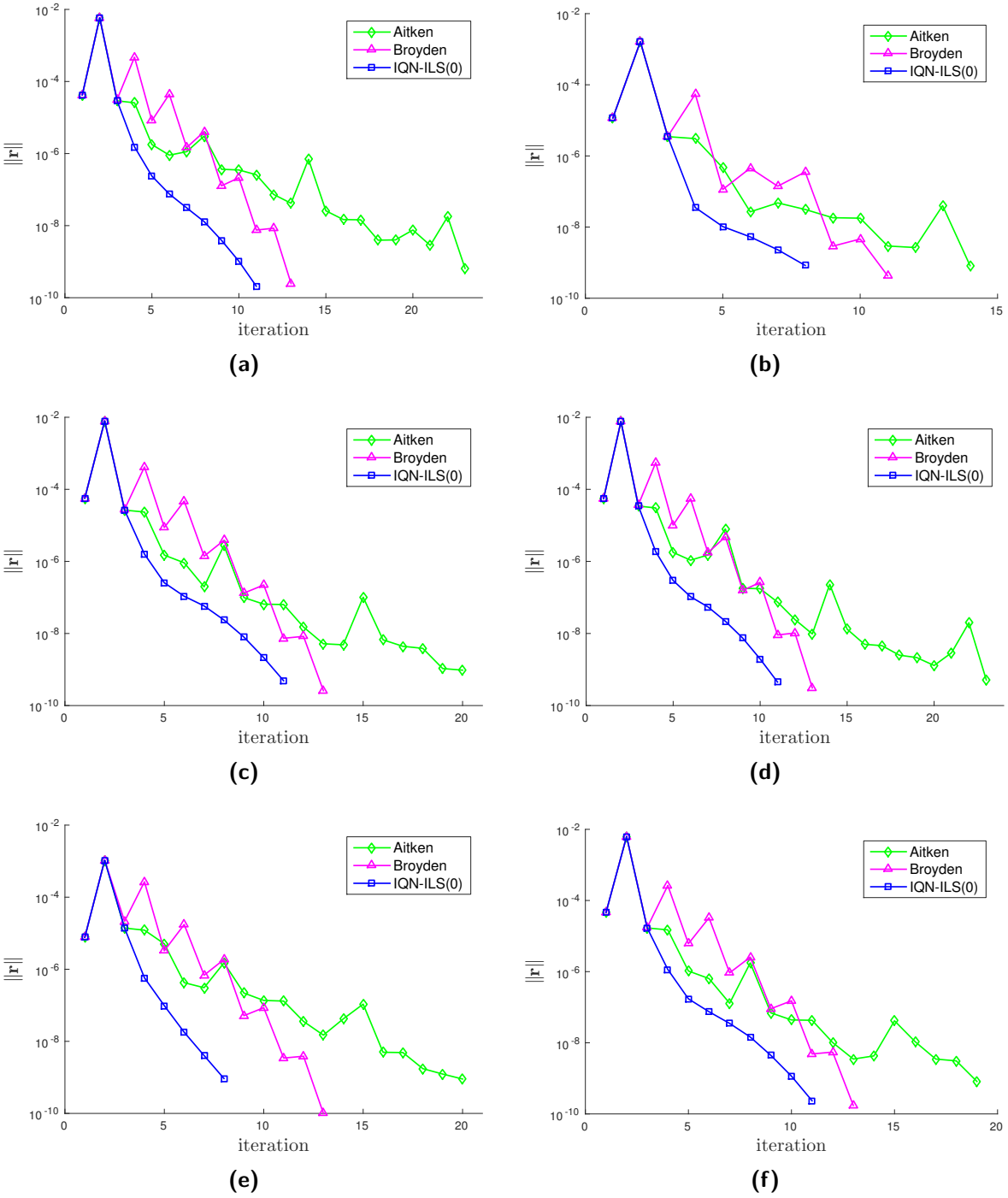


Figure 7.10: Convergence histories for the channel flow problem using several mono-fidelity coupling algorithms at (a) $\bar{t} = 0.15$, (b) $\bar{t} = 0.30$, (c) $\bar{t} = 0.45$, (d) $\bar{t} = 0.60$, (e) $\bar{t} = 0.75$, (f) $\bar{t} = 0.90$, where $\bar{t}$ is a fraction of the period of the periodic inflow.

In Figure 7.10 the residual histories of the channel flow problem is shown for six representative time steps when using the mono-fidelity coupling algorithms Aitken, Broyden and IQN-ILS(0). It is clearly observed that IQN-ILS(0) consistently requires the least number of coupling iterations to converge to the residual norm, $\|\mathbf{r}\| = 10^{-9}$, followed Broyden and that Aitken requires the most.

The residual norms obtained by all three algorithms for the first two iterations are exactly the same, as expected, because the algorithms use the same initial guess and also apply a Gauss Seidel update for the first coupling iteration. Another observation is that the residual norm after the second iteration is consistently higher than the residual norm after the first iteration for all three coupling algorithms.

After the first two iterations, the residual norms obtained by IQN-ILS(0) decreases monotonically in contrast to the residual norms obtained by Aitken and Broyden. The behaviour of the residual histories obtained by Broyden is peculiar. Each decrease in the residual norm is followed by an increase. Fortunately, the decreases are larger than the increases, hence, in spite of the zig-zag behaviour Broyden does converge. Also, the magnitude of the increases appear to decrease for each subsequent increase. One exception is seen in Figure 7.10b, where the residual norm at the 7^{th} iteration is higher than the residual norm at the 5^{th} iteration.

Although, Aitken requires more iterations to converge to $\| \mathbf{r} \| = 10^{-9}$ than Broyden, Aitken initially converges a little bit faster. Therefore, in case of the channel flow problem, Aitken might be a better choice than Broyden if a less strict tolerance is applied.

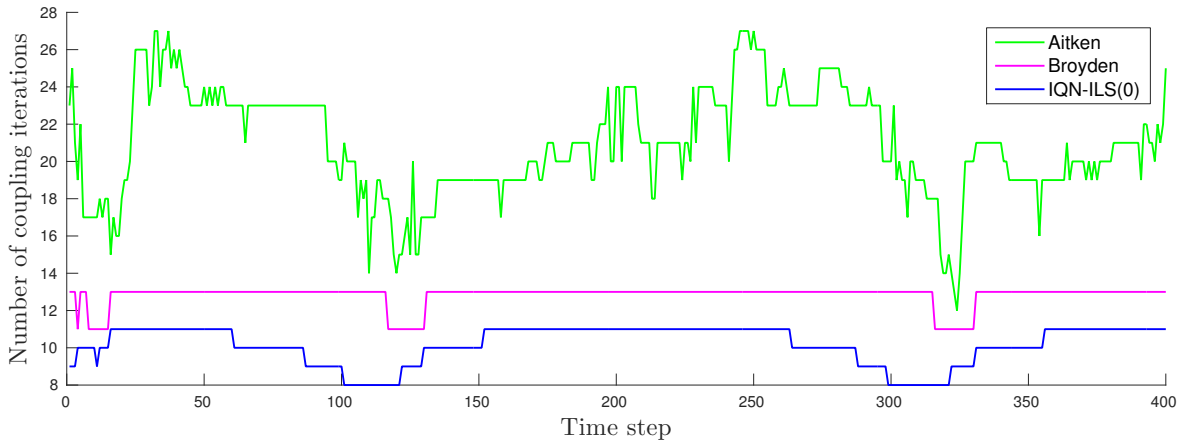


Figure 7.11: Number of coupling iterations required for the convergence of the channel flow problem at each time step for all mono-fidelity coupling algorithms

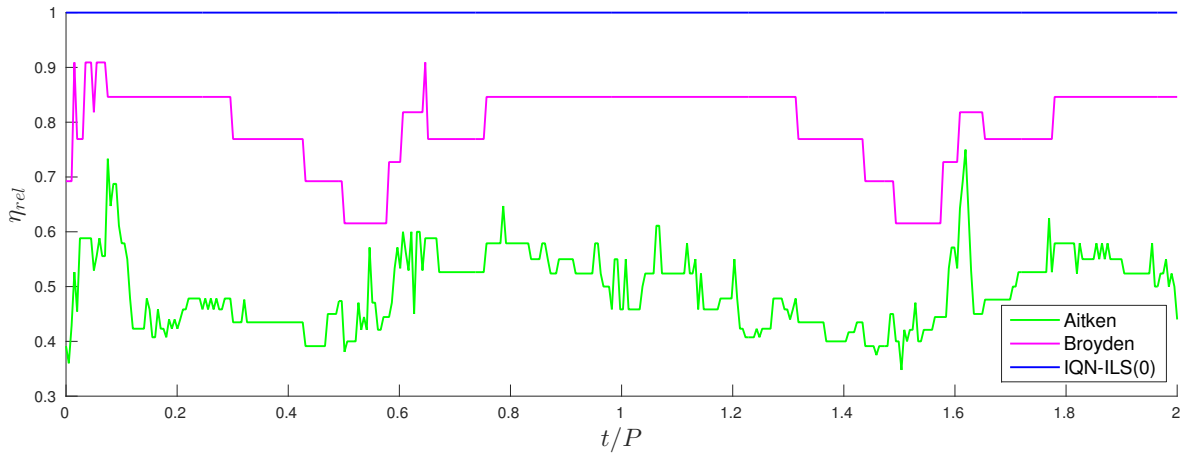


Figure 7.12: Relative efficiency of Aitken and Broyden during the simulation of the channel flow problem when using IQN-ILS(0) as reference

Figure 7.11 depicts the number of coupling iterations per time step required for convergence of the channel flow problem up to a residual norm of $\| \mathbf{r} \| = 10^{-9}$ by the mono-fidelity coupling algorithms during the entire simulation. On average, Aitken, Broyden and IQN-ILS(0) require, respectively, 20.93, 12.81 and 10.17 coupling iterations per time step to converge.

The relative efficiency of Aitken and Broyden compared to IQN-ILS(0) (on basis of the number of iterations) is shown in Figure 7.12 for the entire simulation. On average, Aitken and Broyden have a relative efficiency of 0.49 and 0.79, respectively, when compared to IQN-ILS(0).

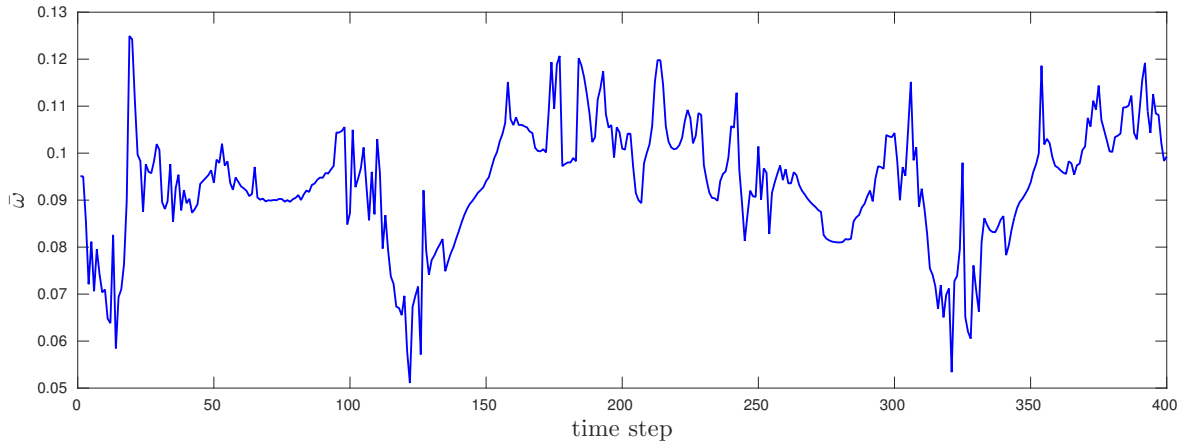
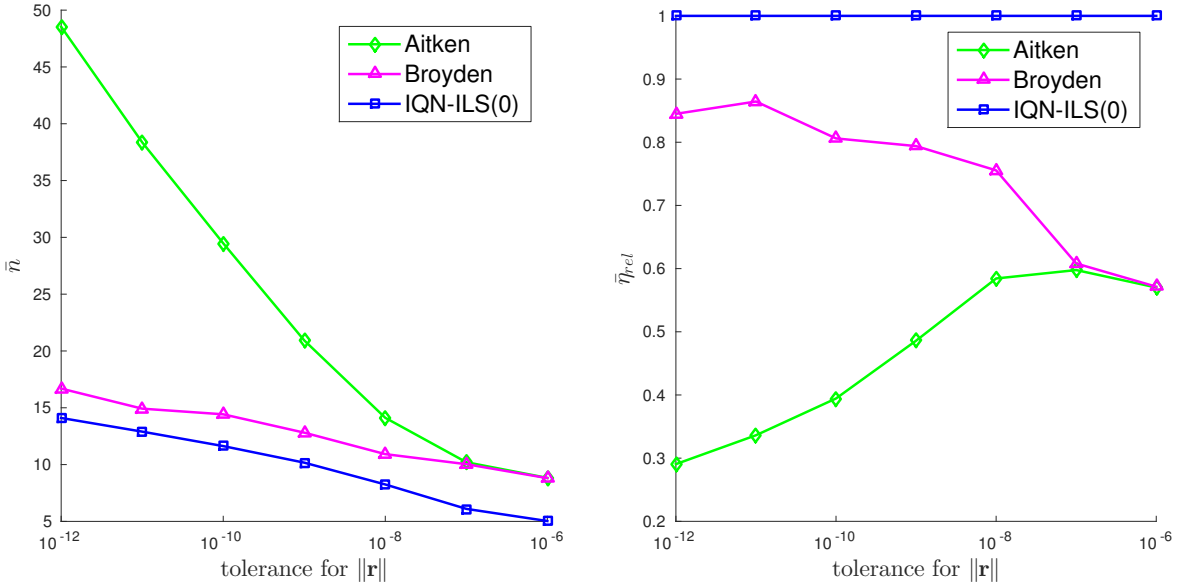


Figure 7.13: The dynamic relaxation factor as computed by Aitken throughout the entire simulation, averaged per time step

Figure 7.13 depicts the dynamic relaxation factor computed by Aitken for the simulation of the channel flow problem up to a residual norm of $\| \mathbf{r} \| \ 10^{-9}$, averaged per time step. As can be seen, the time-step-averaged factor never becomes larger than 0.13. Hence, a considerable amount of relaxation is needed for optimal convergence by fixed-point iteration algorithms. This explains why Gauss Seidel was unable to converge for the channel flow problem as Gauss Seidel is a fixed-point iteration algorithm which does not apply relaxation (i.e. $\omega = 1$).

Influence of residual norm tolerance on coupling iterations



(a) Average number of iterations per time step for various tolerances

(b) Efficiency of coupling algorithms relative to IQN-ILS(0) for various tolerances

Figure 7.14: Comparison of the number of coupling iterations needed by mono-fidelity coupling algorithms to converge the channel flow problem for various tolerances

The influence of the residual tolerance on the performance of Aitken, Broyden and IQN-ILS(0) is depicted in Figure 7.14, where the average number of coupling iterations per time step is shown in Figure 7.14a and the efficiency of Aitken and Broyden relative to IQN-ILS(0) is shown in Figure 7.14b.

IQN-ILS(0) outperforms Aitken and Broyden for the range of tolerance shown. The increase in the average number of coupling iterations when decreasing the tolerance is modest for Broyden and IQN-ILS(0). On the other hand, the increase for Aitken is quite steep. Where the relative efficiency of Broyden increases to about 0.85 as the tolerance becomes smaller, the relative efficiency of Aitken declines to lower than 0.30.

Influence of r in IQN-ILS(r) on number of coupling iterations

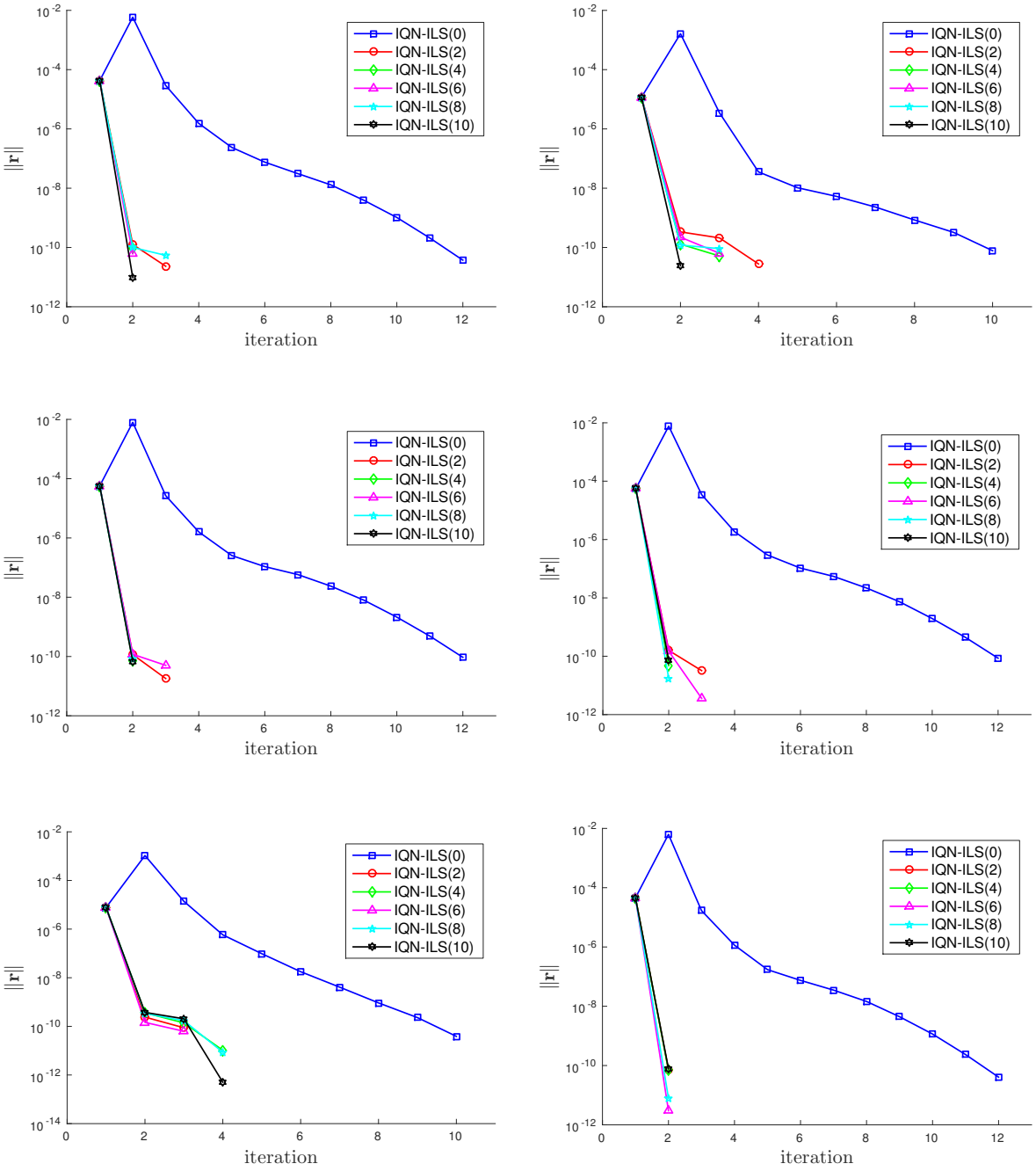


Figure 7.15: Convergence histories for IQN-ILS(r) at (a) $\bar{t} = 0.15$, (b) $\bar{t} = 0.30$, (c) $\bar{t} = 0.45$, (d) $\bar{t} = 0.60$, (e) $\bar{t} = 0.75$, (f) $\bar{t} = 0.90$, where $\bar{t}$ is a fraction of the period of the periodic inflow.

In Figure 7.15 the residual history of IQN-ILS(r) is shown for several numbers of reuse, r, at six representative time steps. The benefit of reusing data from previous time levels is clearly observed when comparing the residual histories for $r > 0$ to the residual history by IQN-ILS(0). Considerably

less coupling iterations are needed for convergence when data from previous time levels is reused. Reductions up to 83% are observed. For $r \geq 6$ IQN-ILS(r) converges oftenly after 2 coupling iterations. Also, when reusing data from previous time levels, the Gauss Seidel update is no longer needed for the first coupling iteration. The benefit is that for $r > 0$ the second residual norm is actually smaller than the first in contrast to what is seen for IQN-ILS(0) in Figure 7.15 and also seen for Aitken and Broyden in Figure 7.10.

The number of coupling iterations per time step needed for convergence is shown for IQN-ILS(r) in Figure 7.16. The average over the entire simulation and the relative efficiency when compared to IQN-ILS(0) is shown in Figure 7.17. From the latter it is observed that for $r = 2$ IQN-ILS(r) already requires on average a factor 2.97 less coupling iterations to converge than IQN-ILS(0). The peak is a factor 4.78 for $r = 8$. The difference in performance for $r = 6$, 8 and 10 is very small.

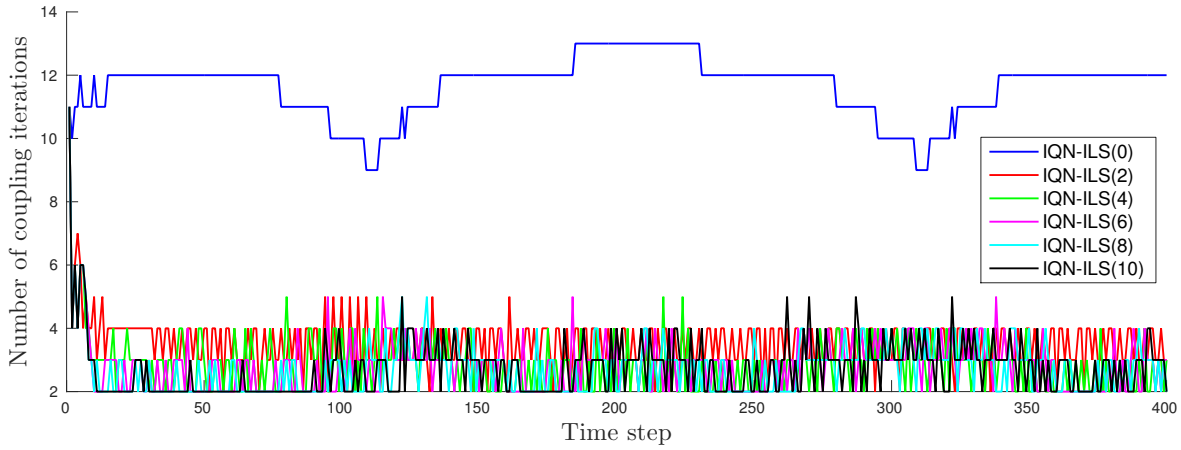
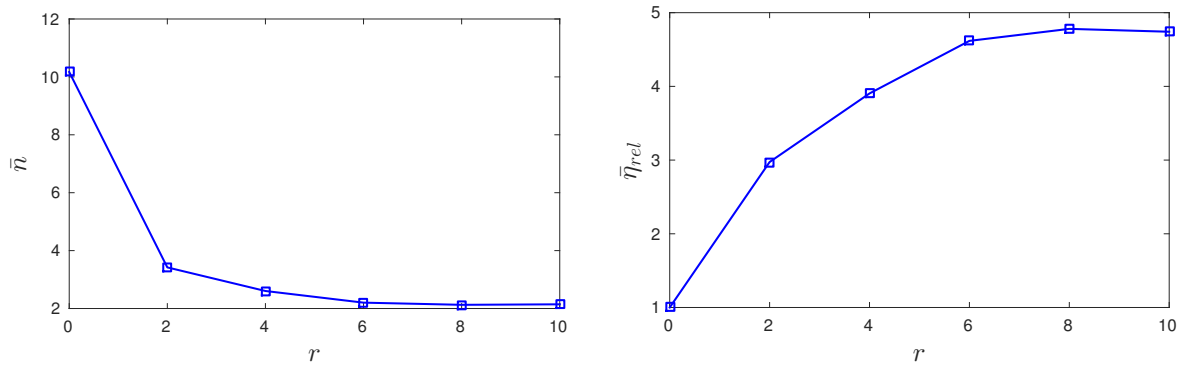


Figure 7.16: Number of coupling iterations required for the convergence of the channel flow problem at each time step for several numbers of reuse by IQN-ILS(r)



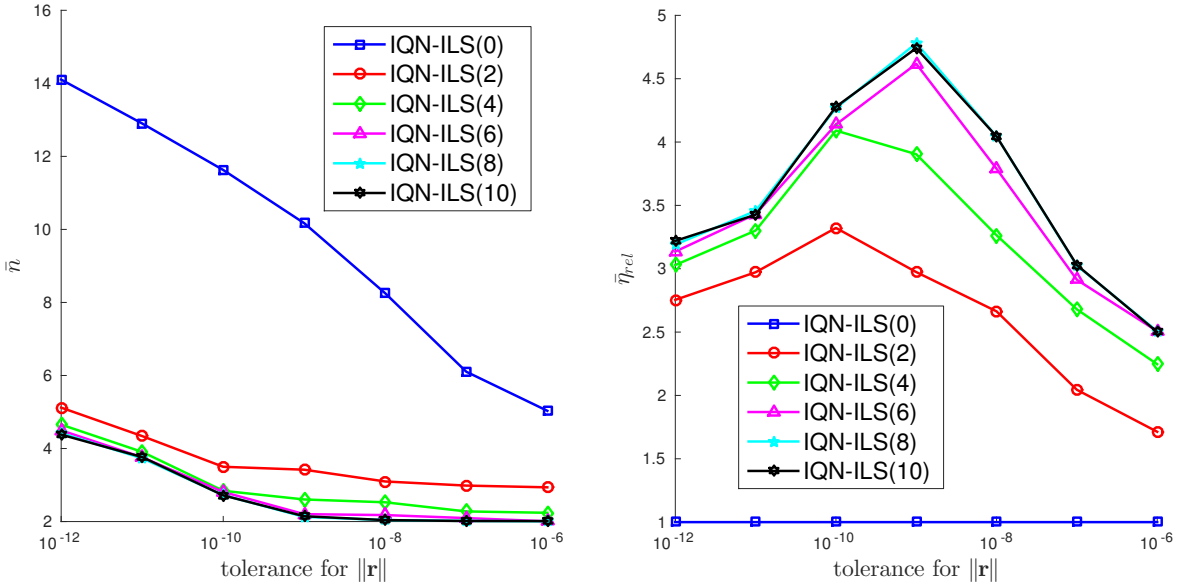
(a) Average number of coupling iterations per time step needed by IQN-ILS(r) for various number of reuse, r

(b) Efficiency of IQN-ILS(r) relative to IQN-ILS(0) for various number of reuse, r

Figure 7.17: Performance comparison of IQN-ILS(r) relative to IQN-ILS(0) for the channel flow problem

The average number of coupling iterations needed by IQN-ILS(r) to converge for various tolerances is shown in Figure 7.18a, the corresponding efficiency relative to IQN-ILS(0) is shown in Figure 7.18. For $r = 2$ and $r = 4$ the relative efficiency increases for $10^{-10} < \| \mathbf{r} \| < 10^{-6}$ and decreases for

$10^{-12} < \| \mathbf{r} \| < 10^{-10}$. Similar behaviour is also seen for $r = 6$, 8 and 10, although, the optimum occurs for these at $\| \mathbf{r} \| = 10^{-9}$.



(a) Average number of iterations per time step for various tolerances

(b) Efficiency of IQN-ILS(r) relative to IQN-ILS(0) for various tolerances

Figure 7.18: Comparison of the number of coupling iterations needed by IQN-ILS(r) to converge the channel flow problem for various tolerances

Discussion of mono-fidelity results

The results presented for mono-fidelity coupling algorithms in this section show that IQN-ILS(0) outperforms Gauss Seidel, Aitken and Broyden when applied for the partitioned coupling of the quasi-one-dimensional channel flow problem. Gauss Seidel is the least performant as it was unable to converge to and caused the residual norm to diverge after a few coupling iterations. The instability of Gauss Seidel is due to its lack of relaxation (i.e. $\omega = 1$). This is further substantiated by inspecting the values Aitken computes for its dynamic relaxation factor. The average throughout the entire simulation and the largest average per time step were 0.09 and 0.12, respectively.

Aitken and Broyden required, respectively, 51% and 21% more coupling iterations than IQN-ILS(0) to converge the channel flow problem up to a residual norm of 10^{-9}. The efficiency of Broyden relative to IQN-ILS(0) increases for decreasing residual norm tolerances, whereas the relative efficiency of Aitken decreases for decreasing residual norm tolerances. The lower performance of Aitken when compared to Broyden and IQN-ILS(0) can be contributed to the large amount of relaxation used by Aitken. Large relaxation implies that a new estimate closely resembles the previous estimate. Even though that has a positive effect on the stability of a fixed-point iteration algorithm, it has a negative effect on the convergence rate.

The residual histories for simulations using Broyden portrayed a zig-zag behaviour where each decrease in residual norm was followed by an increase. This behaviour is not understood, yet. Further investigation is required. Despite of the zig-zag behaviour Broyden was able to converge faster than Aitken.

Results for IQN-ILS(r) show that large performance gains can be achieved, relative to IQN-ILS(0), by reusing data from previous time steps. For converging the channel flow problem up to a residual norm of 10^{-9}, the reduction in the average number of coupling iterations per time step peaked at a

factor of 4.78 when reusing data from 8 previous time steps. A convergence study showed that the efficiency of IQN-ILS(r) relative to IQN-ILS(0) when using $r \geq 6$ increases for $10^{-9} < \|\mathbf{r}\| < 10^{-6}$ and decreases for $10^{-12} < \|\mathbf{r}\| < 10^{-9}$. This is explained by the high convergence rate of IQN-ILS(r). For $r \geq 6$ IQN-ILS(r) converges oftenly to a residual norm close to 10^{-10} after two coupling iterations. Hence, for $10^{-9} < \| \mathbf{r} \| < 10^{-6}$ the average number of coupling iterations needed for convergence does not change for IQN-ILS(r), whereas for IQN-ILS(0) it increases monotonically. By reducing the residual norm tolerance even further IQN-ILS(r) starts to needing more iterations to converge and this explains the drop in relative efficiency for $10^{-12} < \|\mathbf{r}\| < 10^{-10}$.

7.2.2 Multi-fidelity results

Comparison of high- and low-fidelity channel responses

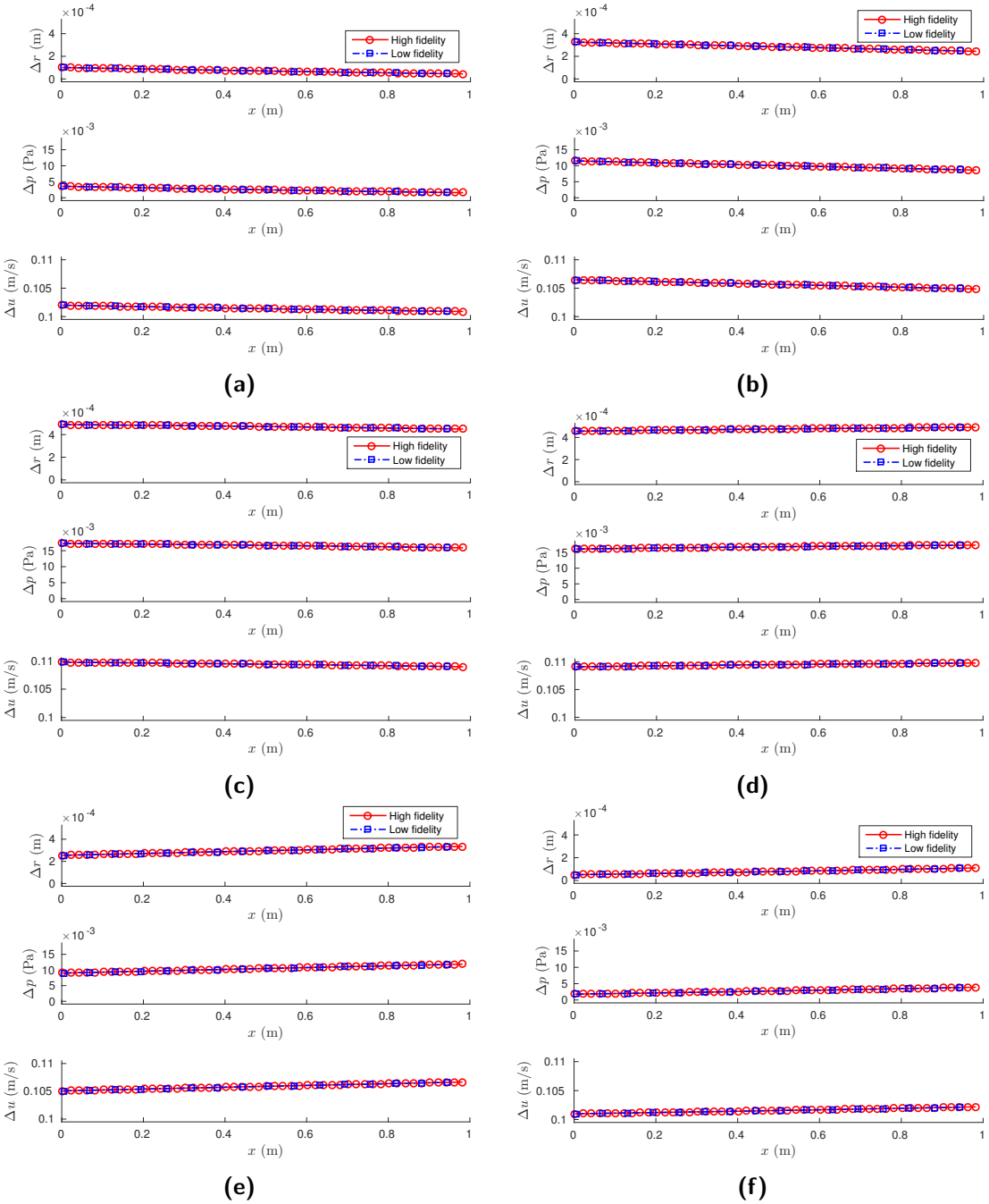


Figure 7.19: Comparison of the high- and low-fidelity model solutions at (a) $\bar{t} = 0.15$, (b) $\bar{t} = 0.30$, (c) $\bar{t} = 0.45$, (d) $\bar{t} = 0.60$, (e) $\bar{t} = 0.75$, (f) $\bar{t} = 0.90$, where $\bar{t}$ is a fraction of the period of the periodic inflow. For visual purposes only a fifth of the data points are shown.

The responses (e.g. radial deformation, fluid pressure and axial flow velocity) of the high-fidelity and low-fidelity models for the quasi-one-dimensional channel flow problem are shown in Figure 7.19 for six representative time levels. For the sake of visual clarity only a fifth of the data points are depicted. Nevertheless, it is clearly observed that the low-fidelity model response matches the high-fidelity model response.

Multigrid mapping operator and smoother

The same flow solver and structure solvers were used for the high-fidelity and low-fidelity models of the channel flow problem. The difference in fidelity comes from using 80 finite-volume cells in the high-fidelity mesh and using 250 finite-volume cells in the low-fidelity mesh. Because the meshes used for the high-fidelity and low-fidelity models do not match at the interface, the definition of the mapping operator must be slightly adjusted by incorporating interpolation. Using h and H to denote the cell lengths of the fine mesh and coarse mesh, respectively, the adjusted mapping operator is defined as

$$\mathcal{P} = I_H^h \cdot \arg\min_{\mathbf{z}} \parallel \tilde{\mathcal{R}}(\mathbf{z}) - I_h^H \cdot \mathcal{R}(\mathbf{x}) \parallel, \tag{7.9}$$

where I_H^h and I_h^H are the prolongation and restriction operators, respectively.

For a perfect mapping $\mathcal{P}(\mathbf{x}^*) = \mathbf{z}^*$, however, the restriction operator filters out high-frequency components of the high-fidelity residual, $\mathbf{r} = \mathcal{R}(\mathbf{x})$, leading to $\mathcal{P}(\hat{\mathbf{x}}) = \mathbf{z}^*$, where $\hat{\mathbf{x}}$ is close to but not identical to $\mathbf{x}^*$. Due to the removal of the high-frequency residual components an ASM-accelerated algorithm cannot eliminate the high-frequency error, $\epsilon = \parallel \hat{\mathbf{x}} - \mathbf{x}^* \parallel$.

In [29] it was shown that standard subiteration provides an adequate smoother for multi-grid methods. A decision criterion was proposed in [19] to automate the application of a smoother. For the decision criterion the high-fidelity residual, $\mathbf{r}$, is split into a high-frequency part, $\mathbf{r}_f$ and a low-frequency part, $\mathbf{r}_c$, such that

$$\mathbf{r} = \mathbf{r}_c + \mathbf{r}_f. \tag{7.10}$$

Performing ASM-accelerated coupling iterations leads to $\parallel \mathbf{r}_c \parallel \rightarrow 0$, while $\parallel \mathbf{r} \parallel \rightarrow \parallel \mathbf{r}_f \parallel$. The ratio of the low-frequency and the high-frequency compononents of the high-fidelity residual is then defined as

$$D = \frac{\parallel \mathbf{r}_c \parallel}{\parallel \mathbf{r}_f \parallel} = \frac{\parallel \mathbf{r}_c \parallel}{\parallel \mathbf{r} - \mathbf{r}_c \parallel}. \tag{7.11}$$

If the low-frequency components are of the same order or less than the high-frequency components (i.e. $D \leq 1$), then CASMIR switches automatically to the Gauss Seidel algorithm. When D grows to larger than 1, CASMIR switches back to the ASM-accelerated algorithm selected in the configuration files.

Influence of multi-grid smoothing of on the performance of multi-fidelity coupling algorithms

Residual histories and the values for D at three time steps for ASM-GS, ASM-Aitken, ASM-Broyden and IQN-ILS(0) are shown in Figure 7.20 for converging the channel flow problem up to $\parallel \mathbf{r} \parallel = 10^{-9}$. The value for D decreases for each subsequent coupling iterations but does not become smaller than unity. The multi-fidelity coupling algorithms therefore converge without needing any smoothing. Also, all the multi-fidelity coupling algorithms converge faster than IQN-ILS(0), indicating that speedups have been achieved. ASM-Aitken and ASM-Broyden converge faster than ASM-GS. The difference

between the residual histories for ASM-Aitken and ASM-Broyden is very small.

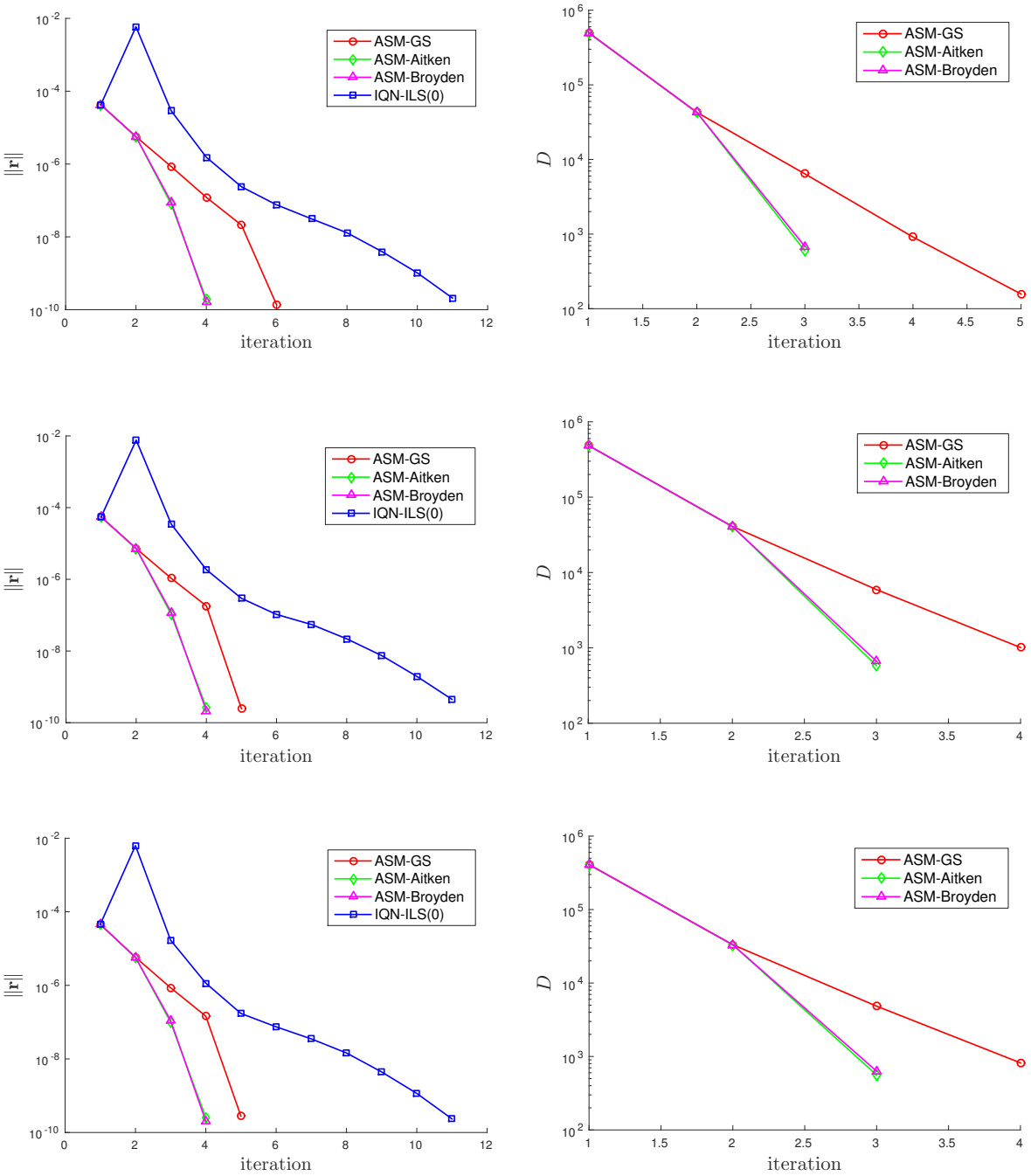


Figure 7.20: Residual histories for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-9}$ on the left and the corresponding D values on the left. $\bar{t} = 0.15$ at the top, $\bar{t} = 0.60$ in the middle and $\bar{t} = 0.90$ at the bottom, where $\bar{t}$ is a fraction of the period of the periodic inflow.

Residual histories and D values for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-9}$ at three time steps are shown in Figure 7.21. In the top subfigures it is observed that both ASM-Aitken and

ASM-Broyden require smoothing for convergence. Where ASM-Aitken converges faster than IQN-ILS(0), ASM-Broyden does not. In the middle subfigures only ASM-Broyden requires smoothing to converge. This time both algorithms converge faster than IQN-ILS(0). In the bottom subfigures ASM-Broyden converges without needing any smoothing, while ASM-Aitken does not converge at all. After smoothing, ASM-Aitken stalls close to the residual norm tolerance and the corresponding D stalls at a value far from unity

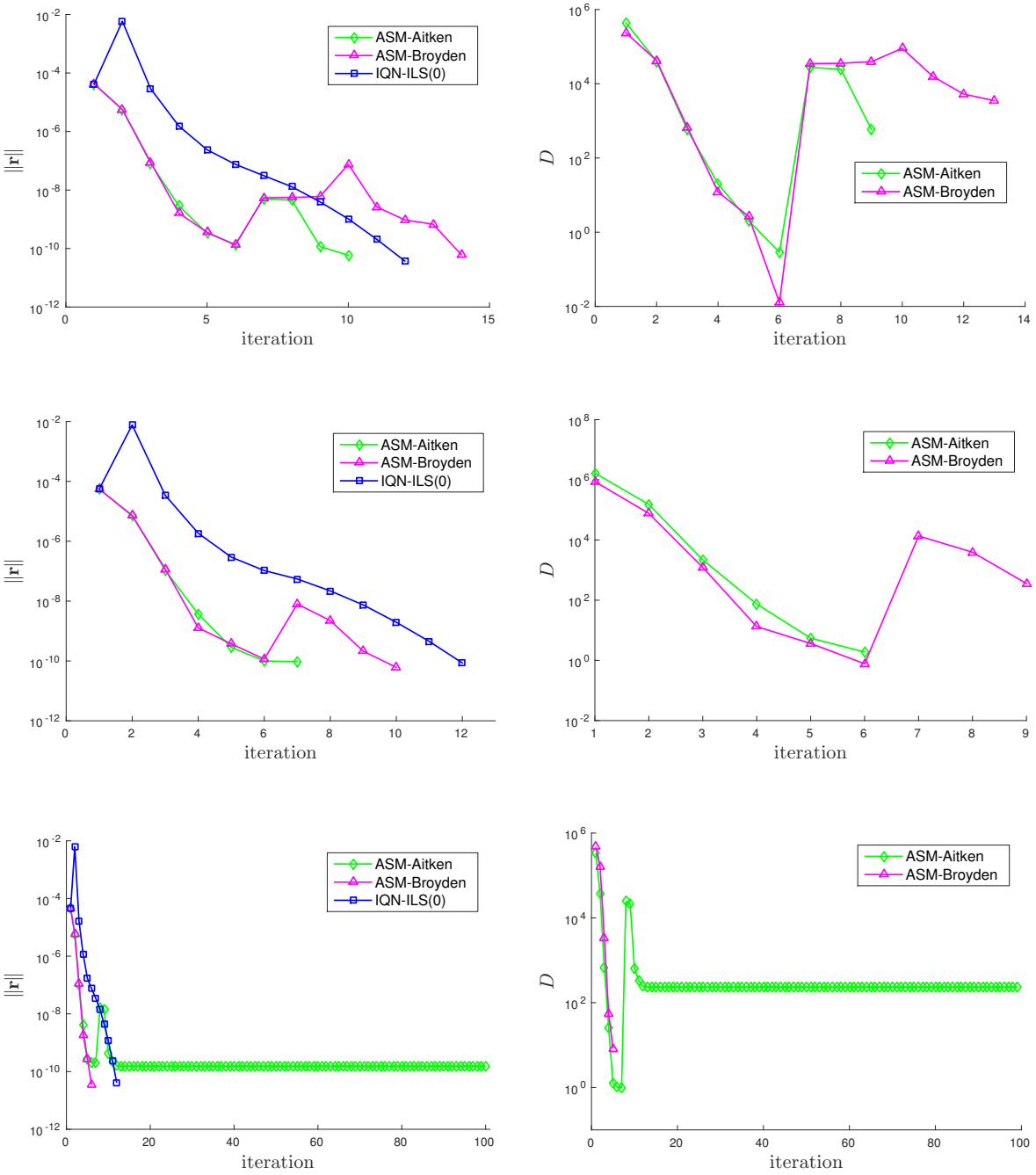


Figure 7.21: Residual histories for converging the channel flow problem up to $\|\mathbf{r}\| = 10^{-10}$ on the left and the corresponding D values on the left. $\bar{t} = 0.15$ at the top, $\bar{t} = 0.60$ in the middle and $\bar{t} = 0.90$ at the bottom, where $\bar{t}$ is a fraction of the period of the periodic inflow.

Comparison of multi-fidelity speedup

Because the residual norms obtained from Gauss Seidel diverged the speedup of ASM-GS relative to Gauss Seidel cannot be determined. The implementation of ASM-ILS appears to perform worse for

the channel problem than for the supersonic panel flutter problem. Some speedup was achieved by ASM-ILS for the supersonic panel flutter, albeit insignificantly small, for the channel flow problem MATLAB spits out errors occuring in low-level built-in functions. In the following results are shown for ASM-GS (where appropriate), ASM-Aitken and ASM-Broyden. Results for ASM-ILS are excluded because more debugging is required in order to provide meaningful results.

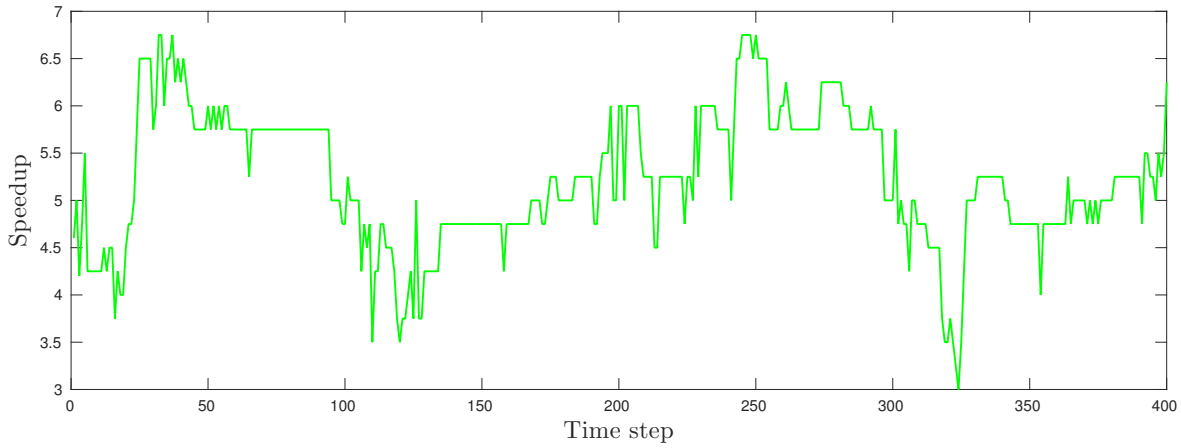


Figure 7.22: Speedup per time step of ASM-Aitken relative to Aitken for converging the channel flow problem up to $\|\mathbf{r}\| = 10^{-9}$

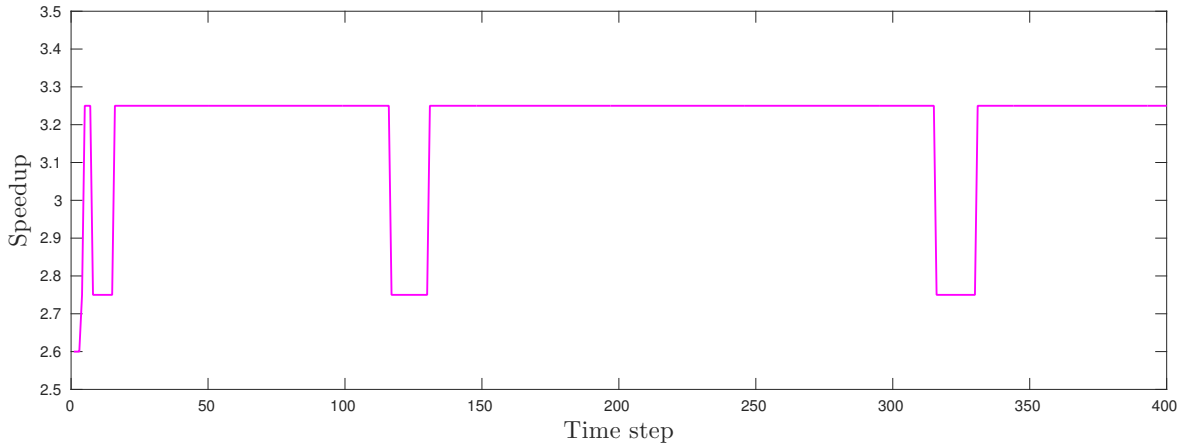


Figure 7.23: Speedup per time step of ASM-Broyden relative to Broyden for converging the channel flow problem up to $\|\mathbf{r}\| = 10^{-9}$

The speedup per time step of ASM-Aitken relative to Aitken for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-9}$ is depicted in Figure 7.22. The minimum, maximum and average speedup are 3.00, 6.75 and 5.22, respectively. The speedup per time step of ASM-Broyden relative to Broyden for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-9}$ is depicted in Figure 7.23. The minimum, maximum and average speedup are 2.60, 3.25 and 3.20, respectively.

To compare the performance of ASM-GS, ASM-Aitken and ASM-Broyden to one another, the speedup per time step must be computed relative to the same basis. Therefore, the speedup relative to IQN-ILS(0) are shown in Figure 7.24. It is observed that, for this particular case, ASM-Aitken and ASM-Broyden perform exactly the same. The minimum, maximum and average speed up per time step are 1.80, 2.75 and 2.54, respectively. Even though Gauss Seidel was unstable ASM-GS is not and

also provides speedup relative to IQN-ILS(0), albeit lower than the speedups obtained by ASM-Aitken and ASM-Broyden. The minimum, maximum and average speedup for ASM-GS are 1.33, 2.20 and 1.94, respectively.

The same type of plot is shown in Figure 7.25 for ASM-Aitken and ASM-Broyden, but now for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-10}$. Two things can be observed, namely, the performance of ASM-Aitken and ASM-Broyden are no longer the same and for many time steps there are speeddowns. The minimum, maximum and average speeddown/speedup of ASM-Aitken relative to IQN-ILS(0) for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-10}$ are 0.10, 2.50 and 1.25, respectively. For ASM-Broyden relative to IQN-ILS(0) the minimum, maximum and average speeddown/speedup are 0.10, 2.40 and 1.36. Even though there are speeddowns for quite a few time steps, ASM-Aitken and ASM-Broyden are on average more efficient than IQN-ILS(0).

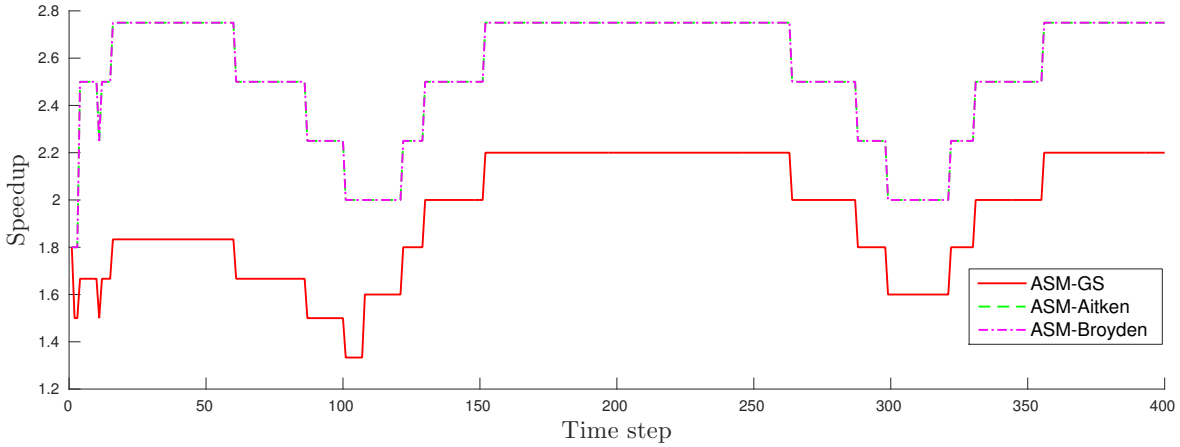


Figure 7.24: Speedup per time step of ASM-Aitken and ASM-Broyden relative to IQN-ILS(0) for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-9}$

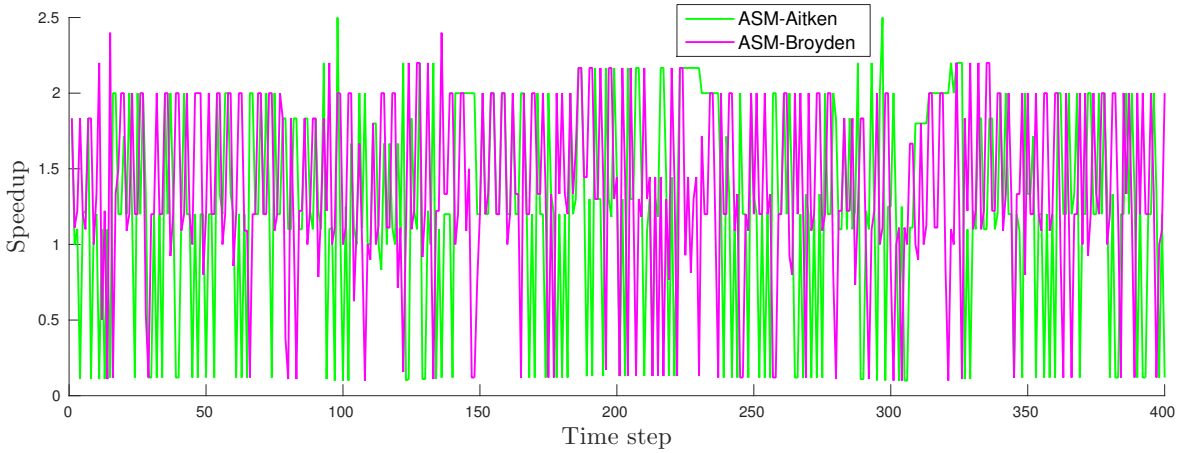


Figure 7.25: Speedup per time step of ASM-Aitken and ASM-Broyden relative to IQN-ILS(0) for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-10}$

Mono-fidelity results for IQN-ILS(r) in Section 7.2.1 showed that a substantial speedup can be gained relative to IQN-ILS(0) from reusing data from previous time levels. It is therefore also important to compare the performance of the ASM-accelerated coupling algorithms to IQN-ILS(r) for various

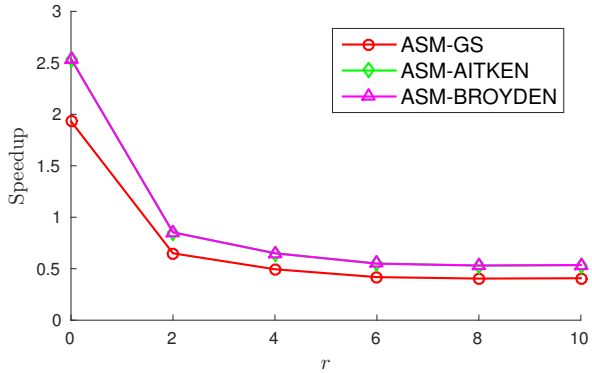


Figure 7.26: Average speedup of multi-fidelity coupling algorithms relative to IQN-ILS(r) for various number of reuse, r, for converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-9}$

number of reuse, r. In Figure 7.26 it is observed that IQN-ILS(r) is more efficient than ASM-GS, ASM-Aitken and ASM-Broyden for $r \geq 2$.

Discussion of multi-fidelity results

The low-fidelity model for the channel flow was derived from the high-fidelity model by using a coarser discretisation of the finite-volume mesh. Physical results for the radial displacement, fluid pressure and flow velocity obtained form the low-fidelity model agreed considerably well with the results from the high-fidelity model.

Because the low-fidelity model used a coarser mesh the mapping operator $\mathcal{P}$ was adjusted accordingly by incorporating restriction of the high-fidelity residual and prolongation of the low-fidelity solution obtained from inner iterations. Multigrid methods often require smoothers due to the removal of high-frequency components through restriction. Similar problems also occured for the ASM-accelerated multi-fidelity coupling algorithms when converging the channel flow problem up to $\| \mathbf{r} \| = 10^{-10}$. Smoothing was employed by means of Gauss Seidel iterations when the low-frequency components of the high-fidelity residual became smaller than the high-frequency components. The usefulness of the smoothing iterations varied among the time steps. Smoothing was not needed in all time steps, but convergence was not always achieved for time steps that needed smoothing. When smoothing was applied and also led to convergence, a speedup relative to IQN-ILS(0) was not always the case. However, on average across the entire simulation, speedups of 1.25 and 1.36 were achieved by ASM-Aitken and ASM-Broden, respectively. Further investigation is needed to explain why convergence was not met for all time steps. Possible causes may be that the ASM formulation is incapable of reducing the residual norm to a very low point or that the tolerance for the inner iterations was not chosen low enough.

The performances of ASM-GS, ASM-Aitken and ASM-Broyden were substantially better when converging to a less strict residual norm tolerance (i.e. $\| \mathbf{r} \| = 10^{-9}$). Smoothing was not needed at all in this case and averaged speedups of 1.94, 2.54 and 2.54 were achieved by ASM-GS, ASM-Aitken and ASM-Broyden, respectively. Both ASM-Aitken and ASM-Broyden are therefore more efficient than ASM-GS, however, the performance of ASM-Aitken and ASM-Broyden did not differ even when comparing the speedup per time step. ASM-Aitken and ASM-Broyden oftenly converged after four outer iterations. Hence, the small difference in performance is explained by the low number of iterations not allowing the residual histories to deviate by much. This is substantiated by the cases converging up to $\| \mathbf{r} \| = 10^{-10}$, where more iterations are needed for convergence. As a consequence the difference between the residual histories of ASM-Aitken and ASM-Broyden become significant after the fourth outer iteration.

Even though ASM-GS, ASM-Aitken and ASM-Broyden achieved speedups relative to IQN-ILS(0),

they proved to be less efficient than IQN-ILS(r) for $r \geq 2$. Hence, reuse of data from previous time steps without applying ASM is more efficient than ASM-GS, ASM-Aitken and ASM-Broyden. Currently, nothing can be said about ASM-ILS due to persisting implementation errors. The expectance is, however, that ASM-ILS(r) will be more efficient than IQN-ILS(r). Further investigation is needed to fix the implementation of ASM-ILS .

Chapter 8

Conclusions and recommendations

This report contains the mathematical derivation, algorithmic fundamentals, design philosophy, implementation overview and performance assessment of CASMIR, a solver-agnostic coupling framework for the partitioned simulation of strongly-coupled fluid-structure interactions (FSI). The goal was to provide a reuseable, flexible and easily extensible tool to accelerate the set up of future partitioned FSI simulations. To this end, CASMIR employs solvers in a black-box fashion (i.e. solver-agnostic), is fully configurable through settings files and incorporates several coupling algorithms based on quasi-Newton iterations. As a means to obtain high-fidelity results from partitioned FSI simulations in a computationally feasible manner, CASMIR also incorporates several multi-fidelity coupling algorithms. These algorithms are based on Aggressive Space Mapping. Two academic test cases, namely, the supersonic panel flutter (linear) and the channel flow (non-linear) problems were used to test CASMIR by assessing the performance of the mono-fidelity and multi-fidelity coupling algorithms.

8.1 Conclusions

8.1.1 Mono-fidelity coupling algorithms

Currently, CASMIR incorporates four mono-fidelity coupling algorithms, namely, Gauss-Seidel, Aitken Δ^2 underrelaxation, Broyden and IQN-ILS(r). The former two are essentially fixed-point iterations methods which were reformulation as a quasi-Newton method to fit into CASMIR's framework, whereas the latter two are actual quasi-Newton methods. The overall conclusions for the mono-fidelity coupling algorithms are

- IQN-ILS(r) is the most efficient mono-fidelity coupling algorithm.
- Always apply IQN-ILS(r) with reuse (i.e. $r > 0$) instead of without reuse (i.e. $r = 0$).
- Prefer Aitken over Broyden for large-scale simulations due to large memory requirements and computational work associated with the latter.
- Avoid Gauss-Seidel and always prefer Aitken if a fixed-point algorithm is to be used.

Gauss-Seidel

The Gauss-Seidel coupling algorithm is very simple and only uses data from the previous coupling iteration to provide a quasi-Newton update. Gauss-Seidel was found to be the least efficient coupling algorithm for the supersonic panel flutter problem and incapable of converging for the channel flow problem. However, when applied for the supersonic panel flutter problem and using relatively small time steps Gauss-Seidel performed similar to the other mono-fidelity coupling algorithms. It is advised to apply Gauss-Seidel only for weakly-coupled FSI problems or when small time steps are used for linear problems.

Aitken

The Aitken coupling algorithm uses dynamic relaxation based on data from two previous coupling iterations. The dynamic relaxation provides improved stability compared to other fixed-point methods such as Gauss-Seidel. Aitken was found to perform marginally better than Gauss-Seidel for the supersonic panel flutter problem and unlike Gauss-Seidel was capable of converging for the channel flow problem. Algorithmically, Aitken is only slightly more complicated than Gauss-Seidel and should therefore always be used instead of Gauss-Seidel. Compared IQN-ILS(0), Aitken was up to 31% less efficient for the supersonic panel flutter and up to 51% less efficient for the channel flow problem.

Broyden

The Broyden coupling algorithm is the only included algorithm which approximates a full inverse-Jacobian. Broyden updates the approximation using data from the previous two coupling iterations. Compared to IQN-ILS(0), Broyden was up to 34% less efficient for the panel flutter problem and up to 31% less efficient for the channel flow problem. Although, Broyden performed better than Gauss-Seidel and Aitken for both test problems its application comes at the cost of needing to store a full inverse-Jacobian in memory and needing a matrix-vector multiplication during each coupling iteration. The storage requirement and computational work associated with Broyden are possible bottlenecks for large-scale simulations. Broyden is, hence, not adviseable for large-scale simulations and Aitken or, more preferably, IQN-ILS(r) should be used instead.

IQN-ILS(r)

The IQN-ILS(r) coupling algorithm uses data from all previous coupling iterations of the current time step and reuses data from up to r previous time steps to provide an accurate quasi-Newton update. IQN-ILS(r) was found to be the most efficient mono-fidelity coupling algorithm for both test problems, even in the case of no reuse (i.e. $r = 0$). The inclusion of data from previous time steps increases the efficiency of IQN-ILS(r) substantially relative to IQN-ILS(0). Through reuse the average number of coupling iterations per time step was reduced by a factor of 2.75 for the supersonic panel flutter problem and a factor 4.78 for the channel flow problem.

8.1.2 Multi-fidelity coupling algorithms

Currently, CASMIR incorporates four multi-fidelity coupling algorithms, namely, ASM-GS, ASM-Aitken, ASM-Broyden and ASM-ILS(r). These algorithms are based on Aggressive Space Mapping, which basically is a reformulated quasi-Newton method that exploits a low-fidelity model to accelerate the iterative convergence of a high-fidelity model. The four ASM variants obtained by incorporating the mono-fidelity coupling algorithms into the framework of Aggressive Space Mapping. The overall conclusions for the multi-fidelity coupling algorithms are

- The implementation of ASM-ILS(r) is faulty and the performance is far below expectance.
- ASM-GS, ASM-Aitken and ASM-Broyden achieve speedups relative to both their respective mono-fidelity counterparts and IQN-ILS(0).
- IQN-ILS(r) with reuse (i.e. $r > 0$) is, however, more efficent than the ASM-GS, ASM-Aitken and ASM-Broyden.

ASM-GS

ASM-GS incorporates a reformulated Gauss-Seidel algorithm to perform outer iterations. Whereas standard Gauss-Seidel was unstable for the channel flow problem, ASM-GS was able to converge for all time steps. ASM-GS was found to be comparable to IQN-ILS(0) for the supersonic panel flutter problem more efficient than IQN-ILS(0) for the channel flow problem by achieving speedups up to 1.94. ASM-GS was, however, less efficient than ASM-Aitken and ASM-Broyden for both test problems.

ASM-Aitken

ASM-Aitken incorporates a reformulated Aitken algorithm to perform outer iterations. ASM-Aitken achieved speedups relative to IQN-ILS(0) up to 1.60 for the supersonic panel flutter problem and up to 2.54 for the channel flow problem. The performance of ASM-Aitken and ASM-Broyden were comparable and did not differ by much.

ASM-Broyden

ASM-Broyden incorporates a reformulated Broyden algorithm to perform outer iterations. Relative to IQN-ILS(0), ASM-Broyden achieved speedups up to 1.5 for the supersonic panel flutter problem and up to 2.54 for the channel flow problem. Due to the same problems as discussed before for standard Broyden it is adviseable to use ASM-Aitken instead of ASM-Broyden for large-scale FSI simulations.

ASM-ILS

ASM-ILS incorporates a reformulated IQN-ILS algorithm to perform outer iterations. The performance of ASM-ILS(0) was extremely poor for the supersonic panel flutter problem. When ASM-ILS converged the performance was at best comparable to that of standard IQN-ILS(0). Slightly better performance was obtained from ASM-ILS(r), yet the performance of ASM-ILS(r) was inferior to that of standard IQN-ILS(r). ASM-ILS(r) was incapable of converging for the channel flow problem due to MATLAB spitting out errors stemming from low-level built-in functions. It is believed that the implementation of ASM-ILS is faulty and not that the ASM-ILS algorithm is a poor coupling algorithm. Further investigation is needed to fix the implementation. The expectance is that ASM-ILS(r) is more efficient than IQN-ILS(r) for equal r.

8.2 Recommendations

The first order of business is to fix the implementation of ASM-ILS(r) and to assess its performance. Thereafter, the interfacing of CASMIR and FLECS needs to be tested properly. Because matching meshes were used for both test problems, FLECS was only used to transfer data between CASMIR and the black-box solvers and not for interpolation. After verifying that the interface between CASMIR and FLECS works properly for simulations where non-matching meshes are involved, the performance

of the coupling algorithms need to be tested on more complex test problems (e.g. 3D problems on large meshes).

Interpolation between large non-matching meshes can be very computationally expensive. This is especially the case for interpolating with Radial Basis Functions. FLECS should therefore be extended with more efficient interpolation methods. Another option is to look into parallelising the implementation of the current interpolation methods.

Coupling algorithms for partitioned simulations are currently heavily researched by several communities. New algorithms which prove to be more efficient than IQN-ILS(r) should be implemented into CASMIR. One can also look into space-mapping methods other than ASM if they prove to be more efficient and incorporate those into CASMIR as well

Bibliography

[1] T. P. Scholcz, A. H. van Zuijlen, and H. Bijl, "A multi-model incremental adaptive strategy to accelerate partitioned fluid-structure interactions using space-mapping," in *proceedings of the IV International Conference on Computational Methods for Coupled Problems in Science and Engineering*, (Kos, Greece), pp. 779–791, 2011.

[2] T. P. Scholcz, A. van Zuijlen, and H. Bijl, "Space-mapping in Fluid-Structure Interaction problems." Submitted to Computer Methods in Applied Mechanics and Engineering, 2012.

[3] L. Florentie, "Acceleration of partitioned fluid-structure interaction simulations by means of space mapping - An analysis of suitable approaches," Master's thesis, Delft University of Technology, 2012.

[4] J. W. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 12, pp. 2536–2544, 1994.

[5] J. W. Bandler, R. M. Biernacki, S. H. Chen, R. H. Hemmers, and K. Madsen, "Electromagnetic optimization exploiting aggressive space mapping," *IEEE Transactions on Microwave Theory and Techniques*, vol. 43, no. 12, pp. 2874–2882, 1995.

[6] C. G. Broyden, "A class of methods for solving nonlinear simulataneous equations," *Mathematics of Computation*, vol. 19, no. 92, pp. 557–593, 1965.

[7] J. Degroote, P. Bruggeman, R. Haelterman, and J. Vierendeels, "Stability of a coupling technique for partitioned solvers in FSI applications," *Computers & Structures*, vol. 86, no. 23-24, pp. 2224–2234, 2008.

[8] J. Degroote, K. J. Bathe, and J. Vierendeels, "Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction," *Computers & Structures*, vol. 87, no. 11-12, pp. 793–801, 2009.

[9] J. Degroote, R. Haelterman, S. Annerel, P. Bruggeman, and J. Vierendeels, "Performance of partitioned procedures in fluid-structure interaction," *Computers & Structures*, vol. 88, no. 7-8, pp. 446–457, 2010.

[10] A. de Boer, A. H. van Zuijlen, and H. Bijl, "Review of coupling methods for non-matching meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 8, pp. 1515–1525, 2007.

[11] H. G. Matthies and J. Steindorf, "Partitioned strong coupling methods for fluid-structure interaction," *Computers & Structures*, vol. 81, no. 8-11, pp. 805–812, 2003.

[12] G. Guidoboni, R. Glowinksi, N. Cavallini, and S. Canic, "Stable loosely-coupled-type algorithm for fluid-structure interaction in blood flow," *Journal of Computational Physics*, vol. 228, no. 18, pp. 6916–6937, 2009.

[13] H. Bijl, A. H. van Zuijlen, A. de Boer, and D. J. Rixen, *Fluid-structure interaction - An introduction to numerical coupled simulation.* Lecture notes. Delft University of Technology, 2008.

[14] S. Badia, F. Nobile, and C. Vergara, "Fluid-structure partitioned procedures based on Robin transmission conditions," *Journal of Computational Physics*, vol. 227, no. 14, pp. 7027–7051, 2008.

[15] M. Heil, "An efficient solver for the fully coupled solution of large-displacement fluid-structure interaction problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 1-2, pp. 1–23, 2004.

[16] G. G. Dahlquist, "A special stability problem for linear multistep methods," *BIT Numerical Mathematics*, vol. 3, no. 1, pp. 27–43, 1963.

[17] B. Gustafsson, *Fundamentals of Scientific Computing.* Texts in Computational Sciences and Engineering, Springer, 2011.

[18] R. Haelterman, J. Degroote, D. Van Heule, and J. Vierendeels, "The quasi-Newton Least Squares method: A new and fast secant method analyzed for linear systems," *SIAM Journal On Numerical Analysis*, vol. 47, no. 3, pp. 2347–2368, 2009.

[19] J. J. Kreeft, M. Weghs, A. H. van Zuijlen, and H. Bijl, "Multi-level and quasi-Newton acceleration for strongly coupled fluid-structure interaction," in *proceedings of the IV International Conference on Computational Methods for Coupled Problems in Science and Engineering*, 2011.

[20] A. Beckert and H. Wendland, "Multivariate interpolation for fluid-structure-interaction problems using radial basis functions," *Aerospace Science and Technology*, vol. 5, no. 2, pp. 125–135, 2001.

[21] U. Kuttler and A. W. Wolfgang, "Fixed-point fluid-structure interaction solvers with dynamic relaxation," *Computational Mechanics*, vol. 43, no. 1, pp. 61–72, 2008.

[22] J. Degroote, S. Annerel, and J. Vierendeels, "Stability analysis of Gauss-Seidel iterations in a partitioned simulation of fluid-structure interaction," *Computers & Structures*, vol. 88, no. 5-6, pp. 263–271, 2010.

[23] A. Kaw and E. E. Kalu, *Numerical Methods with Applications: Abridged.* Authar Kaw, 2010.

[24] A. Griewank, "Broyden Updating, the Good and the Bad!," *Documenta Mathematica*, vol. Optimization Stories, pp. 301–315, 2012.

[25] J. E. Gentle, *Matrix Algebra: Theory, Computations, and Applications in Statistics.* Springer Texts in Statistics, Springer, 2007.

[26] T. D. Robinson, M. S. Eldred, K. E. Wilcox, and R. Haimes, "Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping," *AIAA Journal*, vol. 46, no. 11, pp. 2814–2822, 2008.

[27] S. Piperno and C. Farhat, "Partitioned procedures for the transient solution of coupled aeroelastic problems - Part II: Enery transfer analysis and three-dimensional applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 3147–3170, 2001.

[28] J. Degroote and J. Vierendeels, "Multi-solver algorithms for the partitioned simulation of fluid-structure interaction," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 25-28, pp. 2195–2210, 2011.

[29] E. H. Brummelen, "Partitioned iterative solution method for fluid-structure interaction," *International Journal for Numerical Methods in Fluid*, vol. 65, no. 1, pp. 3–27, 2011.

[30] J. N. Reddy, *An introduction to the finite element method.* McGraw-Hill, 3rd international ed., 2006.

Appendices

A 1D Hermite shape functions

Most introductory courses and books on the Finite Element Method (FEM) present at least a derivation for 1D elements of the Lagrange type. A derivation for elements of the Hermite type are presented less frequently. Hence, a succint derivation is given here for 1D elements of the Hermite type. For a derivation of 1D elements of the Lagrange type or even an introduction into FEM, the reader is referred to [30].

The panel in the supersonic panel flutter problem, as described in Section 6.1.3, is modelled after a beam. The result of discretising the panel into several elements of equal length is depicted in Figure A.1.

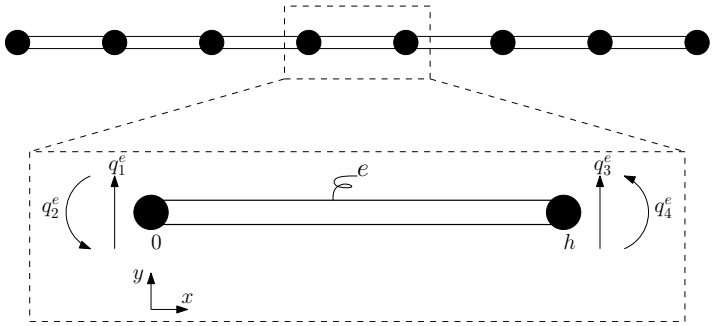


Figure A.1: Representation of a panel modelled after a beam and discretised using beam elements

Because the panel is discretised using 1D beam elements, each node has two degrees of freedom (DOFS). The first is the vertical displacement, w and the second is the slope, $\partial w/\partial x$. A single 1D beam element therefore has four degrees of freedom:

$$\begin{aligned} q_1^e &= w_h^e(0) \\ q_2^e &= \frac{\partial w_h^e}{\partial x}(0) \\ q_3^e &= w_h^e(h) \\ q_4^e &= \frac{\partial w_h^e}{\partial x}(h) \end{aligned} \tag{A.1}$$

The displacement, w_h^e, of an element as function of the local coordinate, x, is assumed to be of the form

$$w_h^e(x) = \sum_{j=1}^{4} q_j^e \psi_j(x), \tag{A.2}$$

where ψ_j are so-called polynomial shape functions. The slope, $\partial w_h^e/\partial x$, of an element as function of the local coordinate, x, is assumed to be of the form

$$\frac{\partial w_h^e}{\partial x}(x) = \sum_{j=1}^{4} q_j^e \frac{\partial \psi_j}{\partial x}(x). \tag{A.3}$$

For the summations in (A.2) and (A.3) to be consistent with (A.1), the shape functions have to satisfy the conditions laid out in Table A.1.

Table A.1: Conditions for deriving Hermite shape functions

	ψ_1	$\frac{\partial\psi_1}{\partial x}$	ψ_2	$\frac{\partial\psi_2}{\partial x}$	ψ_3	$\frac{\partial\psi_3}{\partial x}$	ψ_4	$\frac{\partial\psi_4}{\partial x}$
$x=0$	1	0	0	1	0	0	0	0
$x=h$	0	0	0	0	1	0	0	1

From Table A.1 it follows that each shape function must satisfy four conditions, hence, the shape functions are third order polynomials of the form

$$\psi_i = c_{i0} + c_{i1}x + c_{i2}x^2 + c_{i3}x^3. \tag{A.4}$$

The derivatives of (A.4) have the form

$$\frac{\partial\psi_i}{\partial x} = c_{i1} + 2c_{i2}x + 3c_{i3}x^2. \tag{A.5}$$

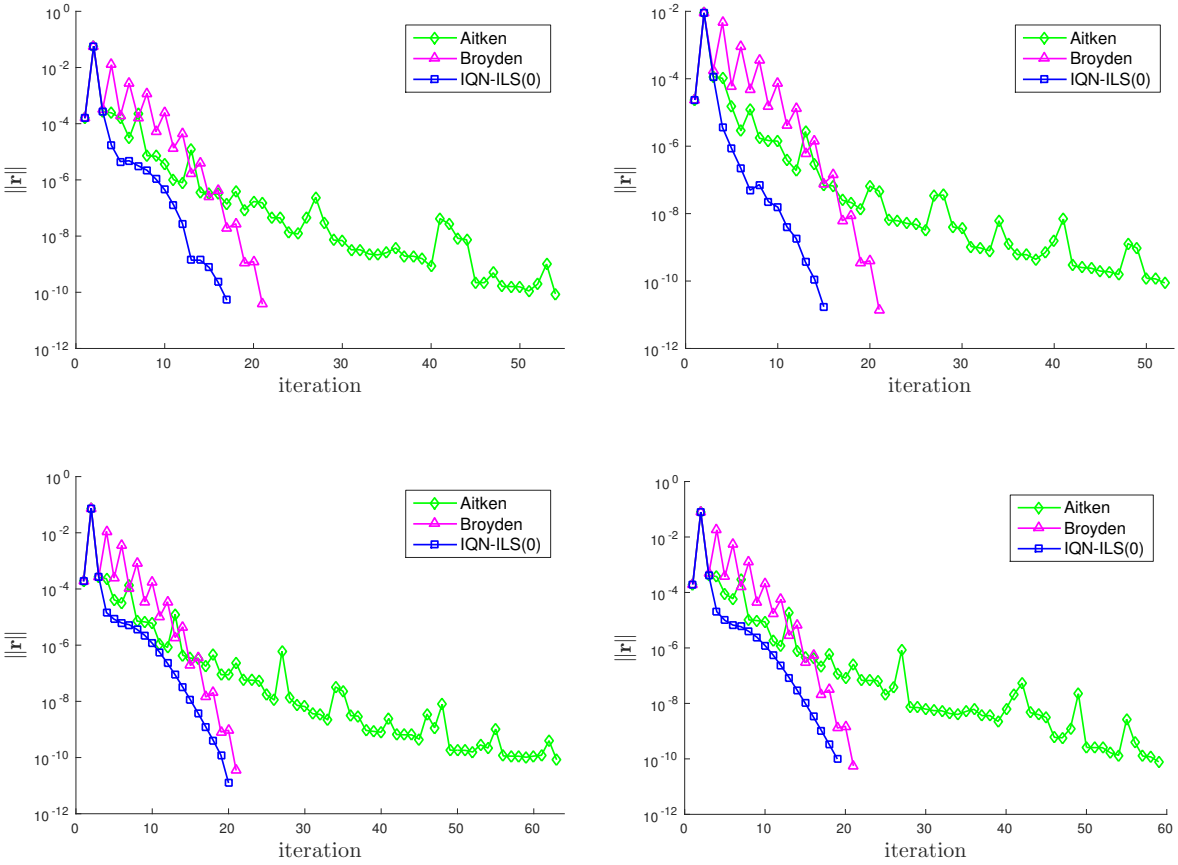
The coefficients c_{ij} are obtained from solving a linear system of equations (established by applying the conditions in Table A.1 on (A.4) and (A.5)),

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & h & h^2 & h^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2h & 3h^2 \end{bmatrix} \begin{bmatrix} c_{10} & c_{20} & c_{30} & c_{40} \\ c_{11} & c_{21} & c_{31} & c_{41} \\ c_{12} & c_{22} & c_{32} & c_{42} \\ c_{13} & c_{23} & c_{33} & c_{43} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{A.6}$$

From (A.6) it is observed that the shape functions need to be determined only once, as long as, all elements have the same length.

B Mono-fidelity channel flow simulations using $\kappa = 10$, $\tau = 0.0025$ and $\epsilon_{FSI} = 10^{-10}$

Influence of coupling algorithm choice on coupling iterations



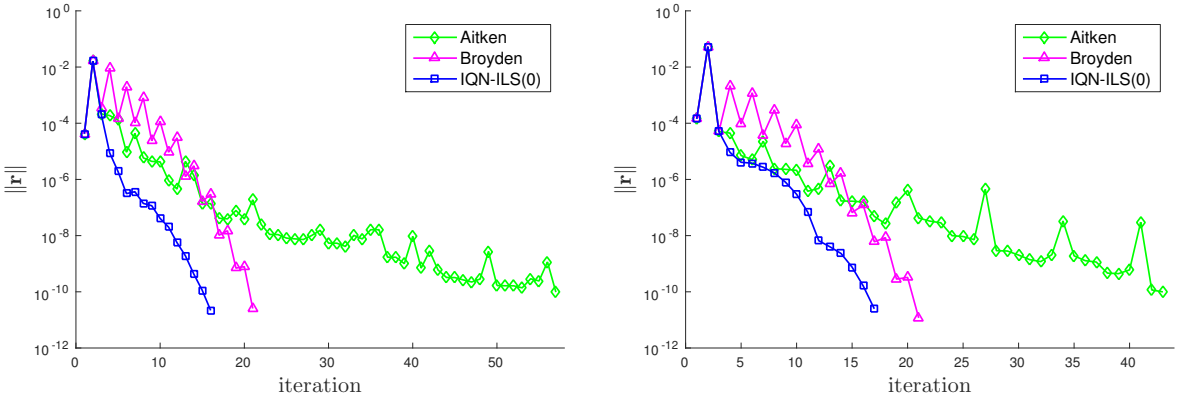


Figure B.2: Convergence histories for several mono-fidelity coupling algorithms at (a) $\bar{t} = 0.15$, (b) $\bar{t} = 0.30$, (c) $\bar{t} = 0.45$, (d) $\bar{t} = 0.60$, (e) $\bar{t} = 0.75$, (f) $\bar{t} = 0.90$, where $\bar{t}$ is a fraction of the period of the periodic inflow.

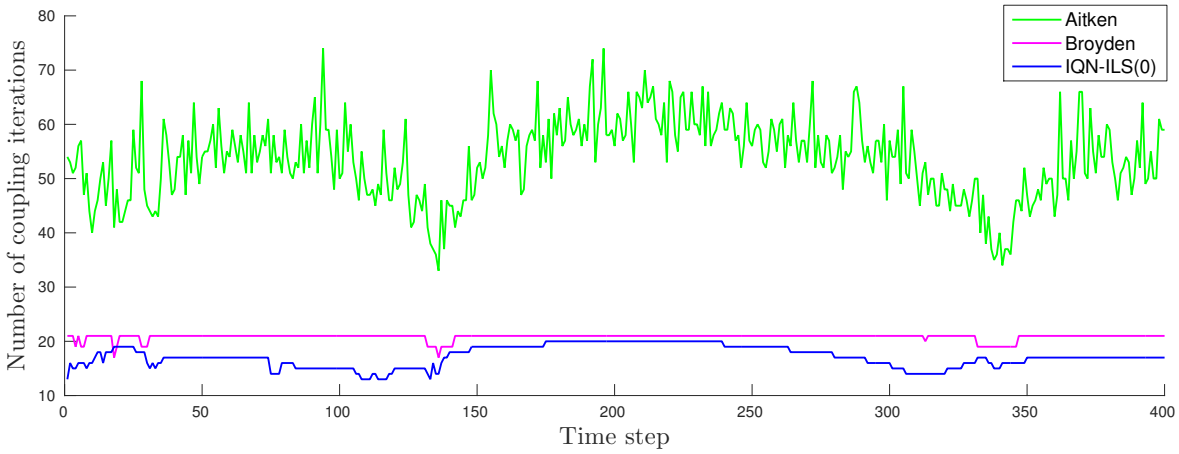
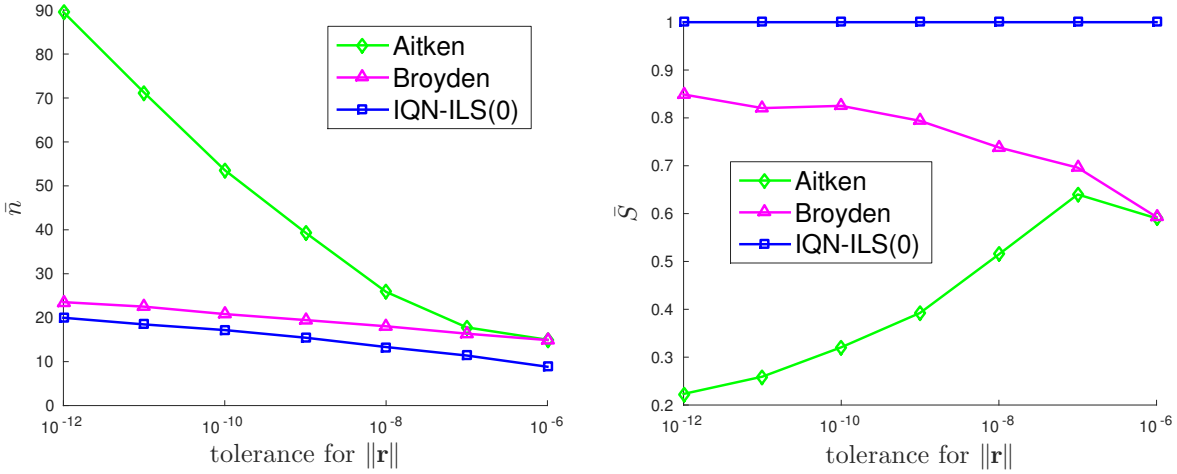


Figure B.3: Number of coupling iterations required for the convergence of the channel flow problem at each time step for all mono-fidelity coupling algorithms

Influence of residual norm tolerance on coupling iterations



(a) Average number of iterations per time step for various tolerances

(b) Efficiency of coupling algorithms relative to IQN-ILS(0) for various tolerances

Figure B.4: Comparison of the number of coupling iterations needed by mono-fidelity coupling algorithms to converge the channel flow problem for various tolerances

Influence of r in IQN-ILS(r) on number of coupling iterations

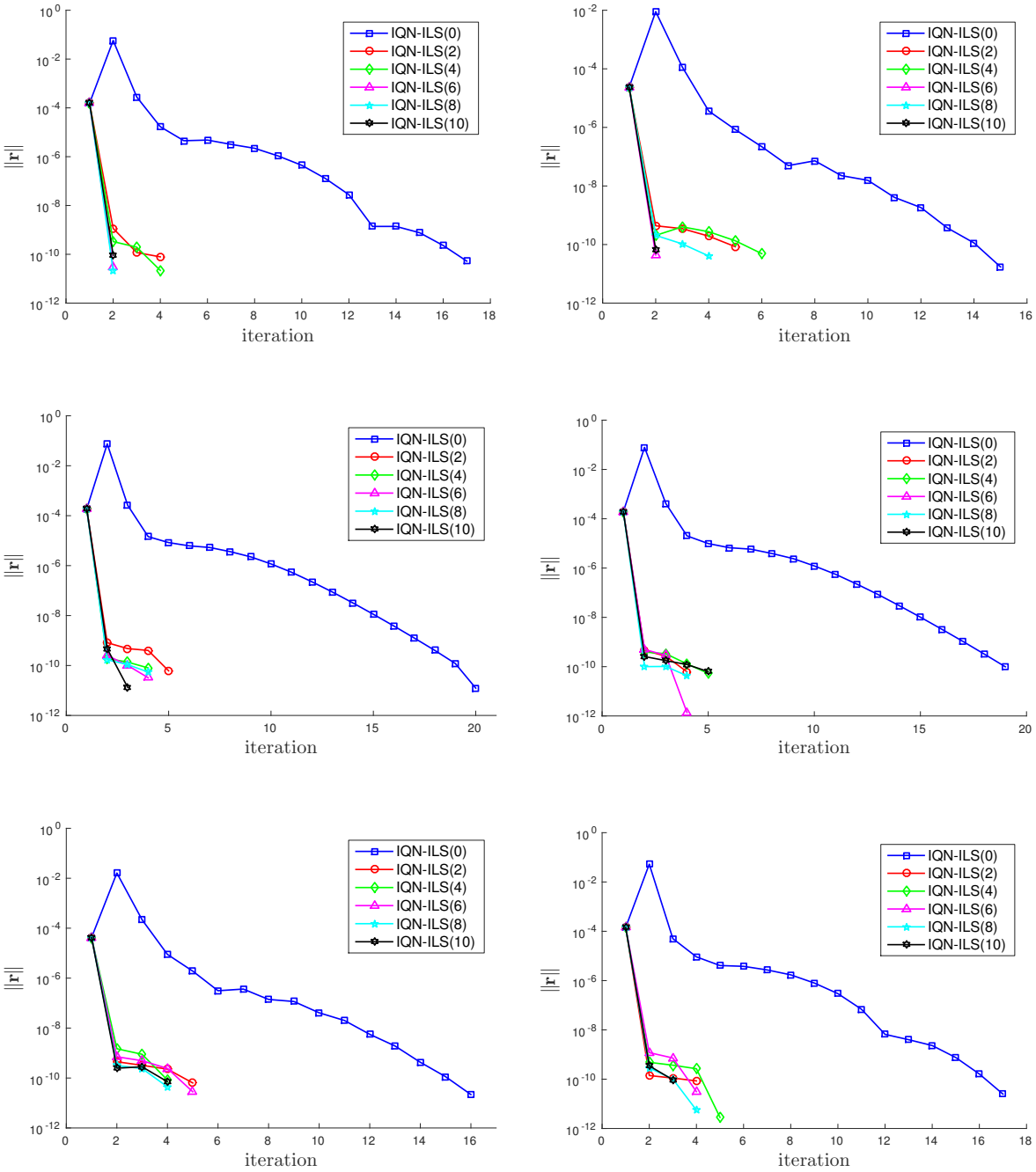


Figure B.5: Convergence histories for IQN-ILS(r) at (a) $\bar{t} = 0.15$, (b) $\bar{t} = 0.30$, (c) $\bar{t} = 0.45$, (d) $\bar{t} = 0.60$, (e) $\bar{t} = 0.75$, (f) $\bar{t} = 0.90$, where $\bar{t}$ is a fraction of the period of the periodic inflow.

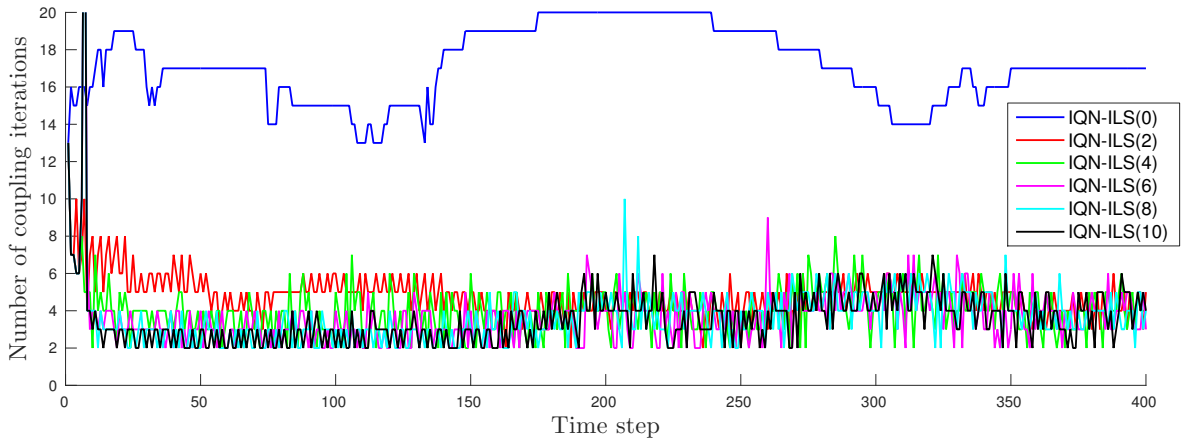
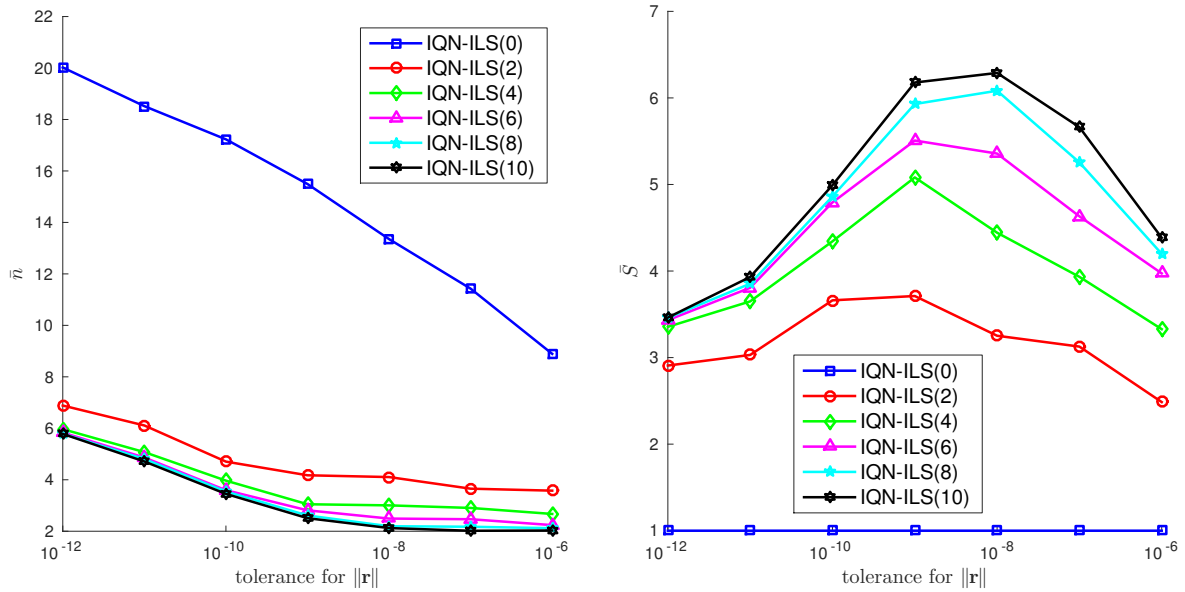


Figure B.6: Number of coupling iterations required for the convergence of the channel flow problem at each time step for several numbers of reuse by IQN-ILS(r)



(a) Average number of iterations per time step for various tolerances

(b) Efficiency of IQN-ILS(r) relative to IQN-ILS(0) for various tolerances

Figure B.7: Comparison of the number of coupling iterations needed by IQN-ILS(r) to converge the channel flow problem for various tolerances