# Improving the Convergence Rate of Rotorcraft Loose Coupling Algorithms using Interaction Laws based on Autoregressive Modelling

Master's Thesis

Maarten Bosmans
University of Twente — National Aerospace Laboratory NLR

August 22, 2011

# Abstract

## Problem area

In many dynamical systems, the physics of the overall system can be modelled as the interaction between several subsystems, each governed by its own laws of physics. A prime example of this is rotorcraft flight, where aerodynamics and structural mechanics are closely coupled. In order to simulate the dynamics of rotorcraft flight, a widely used approach is a Loose Coupling, in which a structural dynamics model and a (computationally expensive) fluid dynamics model are combined with a simplified aerodynamics model. However, even with a loose coupling approach the fluid dynamics model needs to be solved many times before a coupled solution is found. The goal of the present work is to reduce the amount of computational work needed for accurate rotorcraft aeroelastic simulation.

## Description of work

A one-dimensional model problem is constructed that resembles the rotorcraft simulation in a very simple way. It is expected that established techniques from time series analysis can be used to improve the convergence of the coupled models. Using the model problem various methods for improving the loose coupling are evaluated.

## Results and conclusions

The time series resulting from the convergence of the loosely coupled procedure can accurately be modelled as an autoregressive process. This model is subsequently used to estimate the final solution of the iterative procedure, improving the overall convergence rate.

## Applicability

The described methods can be applied to rotorcraft simulations in order to reduce the amount of computational work required or, alternatively, use more accurate models for the same amount of computational work. This Master's Thesis provides a basis for further research into the applicability of time series analysis on rotorcraft loose coupling in particular and partitioned procedures in general.

# Contents

CHAPTER 1

# Introduction

Rotorcraft aerodynamics is considerably more complex than that of fixed wing aircraft. The high aspect ratio and flexibility of the rotor blades result in a complex interplay between the inertial and elastic forces in the structure and the aerodynamic forces acting on the blades. Depending on flight conditions, the wake of the rotor can interact with the fuselage, tail rotor or the rotor itself. An important and characteristic example of such an interaction is known as blade–vortex interaction (BVI). Vortices coming off a rotor blade tip remain in the path of the rotor and interact with one of the next blades. This can be the cause of large force fluctuations on the rotor blade. The resulting blade deformations are the cause of the typical, loud sound produced by a helicopter rotor.

To simulate rotorcraft flight, both computational structural dynamics (CSD) and computational fluid dynamics (CFD) play an important role. An accurate mathematical model of the structure is needed to calculate the blade deformation under load. The CFD model needs to capture aerodynamic properties not only accurately, but also efficiently. The most straightforward way to solve the coupled structural and fluid equations is to combine them into a single formulation. This is called a monolithic scheme. In practice, however, especially in applied rotorcraft research, a monolithic approach is almost never used, because it is difficult to integrate the two different systems of equations in a single numerical procedure. More common are partitioned schemes in which separate CSD and CFD solvers are used to solve the combined system. This approach has advantages in computational efficiency and software modularity, for example because the models and implementations can follow the state of the art in each field separately. Only a monolithic scheme can, however, maintain the conservation properties of the mathematical models it is based on at the CSD-CFD interface. Conservation requires strict compatibility on the approximation spaces at the interface and other conditions on the discretisation, see [1], [2]. A partitioned scheme can, however, be modified to be higher order accurate in energy conservation ([3]).

Because partitioned schemes use separate procedures to solve the fluid and the structural part of the system, the coupling between the two components has to be explicitly defined. Various earlier coupling efforts are described in [4], which provides an extensive review of current rotorcraft CSD and CFD research. In a tightly coupled procedure, the fluid and structure solvers are time-integrated together. Each time step the models exchange information about the state at the fluid–structure boundary. To increase the accuracy for a tight coupling several variations of this basic scheme are possible. For example subcycling the flow solver with a smaller time step or calculating multiple iterations of both models in one time step. Another way of improving overall accuracy is to do the time integration more accurately. In [3] for example, the time marching is done in a staggered approach, where the models are evaluated at interleaved points in time.

Loose coupling removes the requirement of information exchange at each time step. The models are time-integrated separately and are only coupled e.g. once every rotor revolution. The structural model calculates the blade motion for an entire revolution using a simplified model of the aerodynamic forces on the rotor blades. This approximation can come from lifting line theory or from an experimentally found lookup table. The coupling is then achieved by transferring the calculated blade motion to the CFD model and using the computed airflow solution to find better blade loads.

The advantages of a loosely coupled scheme become apparent if one takes rotor trim into account. Because directly comparing experimentally measured trim angles with the simulated results often leads to large discrepancies, the usual approach is to prescribe the total loads on the rotor hub and let the simulation solve for trim. This procedure adds another level of iteration for the convergence of the trimmed coupled system. For tight coupling this results in a procedure that is currently [4] deemed too expensive. With loose coupling, however, the rotor trim can be adjusted during the rotor structural calculations using the simplified aerodynamics model. This is especially useful in the case of steady flight conditions, when the solution is assumed to be periodic.

In the field of financial and statistical modelling a lot of problems are presented as a time series. A time series is a set of data points corresponding to observations of a particular process at different times. In econometrics a time series can for example be the price of a stock at the start of the trading day. Another example application of time series analysis comes from climate research, where the atmospheric history can be studied using the width of tree rings as the time series.

The goal of time series analysis is to extract some properties from the time series that characterise the underlying process. A large and important class of models that is used to describe time series is that of the autoregressive models. These models correlate the current data point with a finite or infinite number of previous points of the series with a linear system of equations. When the model is fitted to all the past observations of the time series, it can be used to predict the future course of the series.

The goal of the present work is to analyse and improve the loose coupling of an aeroelastic rotor simulation for trimmed periodic flight conditions.

The loose coupling of the three models (CFD, CSD and trim) will be formulated as an interaction law, in order to compare it to other uses of interaction laws in the literature. A one-dimensional model problem is constructed that resembles the rotorcraft simulation, including the trimming process, in a very simple way. Using the model problem several methods for improving the loose coupling are proposed. It is expected that established techniques from time series analysis can be used to improve the convergence of the coupled models.

CHAPTER 2

# Interaction laws

Using a partitioned scheme to solve problems involving complex interaction can be challenging with respect to stability and convergence. A possible approach to overcome these problems is the use of an interaction law. An interaction law replaces the boundary condition on the partition boundary in one model with an alternative boundary condition. The new boundary condition approximates the physics of the other model, without actually using the other model. This way the models of which the partitioned scheme is composed of are decoupled. Usually this interaction law is chosen as simple as possible, while retaining enough characteristics of the original model to enhance convergence of the coupled system.

In this chapter the role of the interaction law on a coupled system will be investigated. First some notation will be introduced so that the different interaction laws can be described in a similar manner. Then two examples from the literature are presented in which interaction laws play a role in the convergence behaviour of a coupled system. The second example, in Section 2.3, also includes some analysis about how the interaction law influences the coupled system. Finally, in the last section the concept and notation of an interaction law are applied to the loose coupling approach for rotorcraft aeroelastic modelling.

## 2.1. General formulation

To formalise the concept of an interaction law in the context of coupled systems, a general formulation is given. Let $\mathcal{S}$ and $\mathcal{F}$ be the structural and fluid model with internal state $x$ and $y$ respectively. The two models are coupled by requiring that some set of variables defined for both models have unique values on the interaction surface $\Gamma$. These variables do not necessarily have to be part of the internal state, but should be readily computable from it, using the functions $\mathcal{S}_\Gamma : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_\Gamma}$ and $\mathcal{F}_\Gamma : \mathbb{R}^{n_y} \mapsto \mathbb{R}^{n_\Gamma}$. The combined system can be described as

$$\begin{aligned} \mathcal{S}(x) &= 0 \\ \mathcal{F}(y) &= 0 \\ \mathcal{S}_\Gamma(x) &= \mathcal{F}_\Gamma(y). \end{aligned} \tag{2.1}$$

Instead of finding a solution for this system using a direct method, an iterative method is used. In an iterative procedure the models are alternatingly solved until both are converged to a fixed point. Instead of prescribing all the variables on $\Gamma$ for both models, the set of variables is partitioned into two parts. For one of the parts the values of the variables from $\mathcal{F}_\Gamma$ are used to prescribe the values in $\mathcal{S}_\Gamma$ and conversely, the other part is used as the boundary condition for $\mathcal{F}_\Gamma$.

$$\mathcal{S}_\Gamma(x) = \begin{bmatrix} \mathcal{S}_\Gamma^l(x) \\ \mathcal{S}_\Gamma^r(x) \end{bmatrix} \qquad \mathcal{F}_\Gamma(y) = \begin{bmatrix} \mathcal{F}_\Gamma^r(y) \\ \mathcal{F}_\Gamma^l(y) \end{bmatrix} \qquad \text{with} \quad \substack{\mathcal{S}_\Gamma^l(x), \mathcal{F}_\Gamma^r(y) \in \mathbb{R}^{n_1} \\ \mathcal{S}_\Gamma^r(x), \mathcal{F}_\Gamma^l(y) \in \mathbb{R}^{n_2} \\ n_1 + n_2 = n_\Gamma}$$

The parts are called the left and right part, according to the side on which they appear in the iterative procedure. The iterative procedure is given by

$$\begin{cases} \mathcal{S}(x^n) = 0 \\ \mathcal{S}_\Gamma^l(x^n) = \mathcal{F}_\Gamma^r(y^{n-1}) \end{cases} \\ \begin{cases} \mathcal{F}(y^n) = 0 \\ \mathcal{F}_\Gamma^l(y^n) = \mathcal{S}_\Gamma^r(x^n). \end{cases} \tag{2.2}$$

In this general formulation it is not important how the models $\mathcal{S}$ and $\mathcal{F}$ are solved. In practice an iterative method could very well be used for each of the models, resulting in a nested iterative approach.

In the context of a helicopter rotor aeroelastic simulation, $\mathcal{S}$ could be the CSD model of the rotor blade with $x$ the deflection angles along the blade axis. $\mathcal{F}$ is the CFD model with $y$ the flow state, for example the density, fluid velocity and specific energy for all the grid points. A possible way of coupling the systems is to let $\mathcal{F}_\Gamma^r(y)$ be the aerodynamic pressure of the airflow on the rotor blade and $\mathcal{S}_\Gamma^r(x)$ the position and velocity of the blade. Every iteration of $\mathcal{S}$ then solves the blade deflections using the aerodynamic forces and subsequently $\mathcal{F}$ solves the airflow around the blade with the blade motion prescribed as a boundary condition.

It is important to note that if (2.2) converges, it will converge to a solution of (2.1). For if $y^n = y^{n-1}$ then $\mathcal{S}_\Gamma^l(x^n) = \mathcal{F}_\Gamma^r(y^n)$ and $\mathcal{S}_\Gamma^r(x^n) = \mathcal{F}_\Gamma^l(y^n)$. By construction then also $\mathcal{S}_\Gamma(x^n) = \mathcal{F}_\Gamma(y^n)$.

### 2.1.1. Introducing the interaction law

An interaction law emerges from the observation that if $\mathcal{S}_\Gamma^l(x^n)$ could be solved with $\mathcal{F}_\Gamma^r(y^n)$ instead of $\mathcal{F}_\Gamma^r(y^{n-1})$, the procedure would converge in one step. However, solving $\mathcal{F}_\Gamma^r(y^n)$ together with $\mathcal{S}_\Gamma^l(x^n)$ reduces the iterative procedure to a direct method again, negating the advantage in computational complexity of the iterative method. So instead an approximation $\tilde{\mathcal{F}}_\Gamma^r$ is used that approximates the behaviour of $\mathcal{F}_\Gamma^r(y^n)$ using only $y^{n-1}$ (and possibly $x^n$). The new coupling equation resulting from the use of the approximation is called the interaction law.

A possible interaction law could be based on a linearisation of $\mathcal{F}_\Gamma^r$ around $y^{n-1}$

$$\mathcal{F}_\Gamma^r(y^n) \approx \mathcal{F}_\Gamma^r(y^{n-1}) + \left.\frac{\partial \mathcal{F}_\Gamma^r}{\partial y}\right|_{y^{n-1}} \left(y^n - y^{n-1}\right). \tag{2.3}$$

This formulation still contains $y^n$, so it is not suitable yet to be used in (2.2). Linearisation of $\mathcal{F}_\Gamma^l$ around $y^{n-1}$ and applying the result to $y^n$ yields

$$\mathcal{F}_\Gamma^l(y^n) = \mathcal{F}_\Gamma^l(y^{n-1}) + \left.\frac{\partial \mathcal{F}_\Gamma^l}{\partial y}\right|_{y^{n-1}} (y^n - y^{n-1}).$$

This equation can be rewritten using the relation $\mathcal{F}_\Gamma^l(y^n) = \mathcal{S}_\Gamma^r(x^n)$ to

$$\mathcal{S}_\Gamma^r(x^n) - \mathcal{S}_\Gamma^r(x^{n-1}) = \left.\frac{\partial \mathcal{F}_\Gamma^l}{\partial y}\right|_{y^{n-1}} (y^n - y^{n-1}).$$

Using this equation in (2.3) requires inverting the Jacobian of $\mathcal{F}_\Gamma^l$ and yields

$$\mathcal{F}_\Gamma^r(y^n) \approx \mathcal{F}_\Gamma^r(y^{n-1}) + \left[\frac{\partial \mathcal{F}_\Gamma^r}{\partial y}\left(\frac{\partial \mathcal{F}_\Gamma^l}{\partial y}\right)^{-1}\right]_{y^{n-1}} \left(\mathcal{S}_\Gamma^r(x^n) - \mathcal{S}_\Gamma^r(x^{n-1})\right). \tag{2.4}$$

This approximation of $\mathcal{F}_\Gamma^r(y^n)$ could already be used to replace $\mathcal{F}_\Gamma^r(y^{n-1})$ in (2.2), because it only depends on known variables ($y^{n-1}$ and $x^{n-1}$) and on the state of the model that is to be solved ($x^n$). To simplify the calculation of the interaction law, a simple model is introduced.

### Using a simple model

The Jacobian matrices in (2.4) can be very large, because they both involve $y$, the complete internal state of $\mathcal{F}$. Furthermore the second Jacobian is generally not invertible, because the state $y$ contains much more elements than the boundary conditions for $\mathcal{F}$. However, the product of the Jacobian and the inverse Jacobian is a much smaller matrix. The idea of the interaction law is that this smaller matrix does not need to be computed directly, but can be replaced by a similar matrix and the advantages of better convergence still hold.

This is where the simple model is introduced, as an approximation of the combined Jacobians. As said, the simple model need not be the exact solution of the combined Jacobians, but can be anything with the same general behaviour. Often it is derived from some sort of physical interpretation, using a very simple representation, which can be easily (in a direct way) calculated. This simple model $\tilde{\mathcal{F}}$ thus can be seen as a map from the boundary condition of $\mathcal{F}$ to the boundary condition of $\mathcal{S}$. The complete interaction law to be used as a replacement for $\mathcal{F}_\Gamma^r$ in (2.2) is now

$$\tilde{\mathcal{F}}_\Gamma^r = \mathcal{F}_\Gamma^r(y^{n-1}) + \tilde{\mathcal{F}}\left(\mathcal{S}_\Gamma^r(x^n) - \mathcal{S}_\Gamma^r(x^{n-1})\right). \tag{2.5}$$

When this interaction law is used in (2.2) instead of $\mathcal{F}_\Gamma^r$ the boundary condition for $\mathcal{S}$ is changed. In the case that $\tilde{\mathcal{F}}$ is linear the new boundary condition can be written with all the unknown terms on the left hand side.

$$\mathcal{S}_\Gamma^l(x^n) - \tilde{\mathcal{F}}\left(\mathcal{S}_\Gamma^r(x^n)\right) = \mathcal{F}_\Gamma^r(y^{n-1}) - \tilde{\mathcal{F}}\left(\mathcal{S}_\Gamma^r(x^{n-1})\right)$$

The boundary condition is now more complex, with an extra term containing $x^n$ on the left hand side of the equation. If $\mathcal{S}$ itself is solved using an iterative procedure this means that the simple model $\tilde{\mathcal{F}}$ has to be calculated many times. So each $\mathcal{S}$–$\mathcal{F}$ iteration becomes more computationally expensive. But as the simple model should have the same general behaviour as $\mathcal{F}$, it is expected that there are less $\mathcal{S}$–$\mathcal{F}$ outer iterations necessary.

A desirable property of an interaction law is that although it may change the convergence path, the solution of the converged system is the same as the solution of the original coupled system. For the interaction law as formulated in (2.5) this property holds if $\tilde{\mathcal{F}}$ is linear and $S_\Gamma^r$ is continuous. It can be seen that this is the case by noting that when $x^n \to \bar{x}$ then also $S_\Gamma^r(x^n) \to S_\Gamma^r(\bar{x})$ under continuity of $S_\Gamma^r$. The contribution of $\tilde{\mathcal{F}}$ in (2.5) then vanishes due to its linearity. So if the new procedure converges, the approximated $\tilde{\mathcal{F}}_\Gamma^r$ is exactly equal to the original $\mathcal{F}_\Gamma^r$ and the new procedure converges to the solution of the original partitioned system, regardless of the choice for $\tilde{\mathcal{F}}$. Of course the choice of the simple model does influence the convergence of the iterative procedure. If the simple model does not reflect the behaviour of the original model $\mathcal{F}$, the iterative method will not converge at all.

The interaction law detailed above is only a example. A variation of this interaction law that will be used later is

$$\tilde{\mathcal{F}}_\Gamma^r = \mathcal{F}_\Gamma^r(y^{n-1}) + \tilde{\mathcal{F}}\left(\mathcal{S}_\Gamma^r(x^n)\right) - \tilde{\mathcal{F}}\left(\mathcal{S}_\Gamma^r(x^{n-1})\right). \tag{2.6}$$

This interaction law loosens the requirement on $\tilde{\mathcal{F}}$ and only requires it to be continuous.

## 2.2. Fluid–structure interaction with heat transfer

Interaction laws can be used in all sorts of problems involving partitioned procedures. For example in [5] passenger comfort in an aircraft cabin is modelled. This is done by coupling a human body thermoregulation model, an airflow model and a radiation model. The thermoregulation model $\mathcal{T}$ and the flow model $\mathcal{F}$ are coupled by the requirement that the temperature $T$ and heat flow $q$ on the body surface are the same for both models. The models and their coupling can be described as

$$\begin{aligned} \mathcal{T}(x) &= 0 \\ \mathcal{F}(y) &= 0 \\ \mathcal{T}_\Gamma(x) &= \mathcal{F}_\Gamma(y) \\ \text{with } \mathcal{T}_\Gamma(x) &= \begin{bmatrix} q_b \\ T_b \end{bmatrix}, \ \mathcal{F}_\Gamma(y) = \begin{bmatrix} q_f \\ T_f \end{bmatrix}. \end{aligned} \tag{2.7}$$

The vectors $T$ and $q$ are of equal length and the models $\mathcal{T}$ and $\mathcal{F}$ are well-posed if exactly one of these vectors is prescribed. The coupled system can be solved with an iterative procedure by specifying $q_b = q_f$ as a boundary condition for $\mathcal{T}$ and $T_f = T_b$ for $\mathcal{F}$.

$$\begin{aligned} \mathcal{T}_\Gamma^l(x^n) &= q_b^n & \qquad \mathcal{F}_\Gamma^r(y^{n-1}) &= q_f^{n-1} \\ \mathcal{F}_\Gamma^l(y^n) &= T_f^n & \qquad \mathcal{T}_\Gamma^r(x^n) &= T_b^n \end{aligned}$$

Of course the role of temperature and heat flux could also be reversed. Both these approaches were tried in [5], but it was found that neither resulted in a stable procedure.

The seemingly arbitrary distinction between the temperature as a boundary condition for one model and the heat flow for the other suggests that it might be fruitful to use an approach which is symmetric with respect to the variables used in the coupling. To this end an interaction law is introduced for both the models. The interaction laws use a simple model to characterise the other system. Each simple model is based on a linearisation using a heat-transfer coefficient $h$ to characterise the system it represents.

$$\begin{aligned} q_b &= h_b(T_0 - T_b) \\ q_f &= h_f(T_f - T_\infty), \end{aligned} \tag{2.8}$$

with $T_0$ and $T_\infty$ suitable reference temperatures in the body and airflow respectively. Now instead of using $\mathcal{T}_\Gamma^l(x^n) = q_f^{n-1}$ as the boundary condition for $\mathcal{T}$, (2.8) is used to find the estimate $\tilde{q}_f^n$ using some fixed $h_f$. As the heat-transfer coefficient $h_f$ is not known yet for the current iteration, $h_f^{n-1}$ is used as an approximation. Also $T_f^n$ is not known yet, but from boundary condition for

$\mathcal{F}^l_\Gamma$ it can be seen that it must be equal to $T^n_b$. This leads to the revised boundary condition

$$\begin{aligned}\tilde{\mathcal{F}}^r_\Gamma &= \tilde{q}^n_f \\ &= h_f(T^n_f - T_\infty) \\ &= h^{n-1}_f(T^n_b - T_\infty).\end{aligned}$$

Similarly, the flow model is solved using the new boundary condition

$$\tilde{\mathcal{T}}^r_\Gamma = T_0 - \frac{q^n_f}{h^n_b}.$$

Using (2.8) to eliminate $h^{n-1}$ from these equations, the revised boundary conditions of the interaction law emerge

$$\begin{aligned}\tilde{\mathcal{F}}^r_\Gamma &= q^{n-1}_f \frac{T^n_b - T_\infty}{T^{n-1}_f - T_\infty} \\ \tilde{\mathcal{T}}^r_\Gamma &= T_0 + \frac{q^n_f}{q^n_b}\left(T^n_b - T_0\right).\end{aligned}$$

In [5] the original system (2.7) is solved using these revised boundary conditions. The interaction law stabilises the system and the iterative procedure now converges.

To see why the coupling between the two models is now symmetric, the coupling equations can be rewritten. From the form in which they appear in the original paper,

$$\begin{aligned}q^n_b &= \left(T^n_b - T_\infty\right) h^{n-1}_f \\ q^n_f &= \left(T_0 - T^n_f\right) h^n_b,\end{aligned}$$

it is clear that with the application of the interaction law the arbitrary distinction between the prescribed variables on the model boundaries is removed. The physical interpretation of this is that the Dirichlet and Neumann boundary conditions in the original system are replaced by a mixed boundary condition for both models.

## 2.3. Viscous boundary layer interaction

Another example of an interaction law involves the viscous–inviscid interaction described by Veldman in [6] and [7]. In an attempt to reduce the computational effort of simulating viscous airflow around an airfoil, viscous flow is only calculated in a small layer around the airfoil. In the flow domain outside the viscous layer, the airflow is assumed to be inviscid. The models are coupled by requiring the density and flow speed of both models to be the same on the interface between the viscous and inviscid flow, $e$.

In the viscous region the boundary-layer equations are used, which describe a thin boundary layer around the airfoil. The influence of the boundary layer on the inviscid airflow is modelled by a displacement effect, i.e. the airfoil appears thicker due to viscous effects. This displacement $\delta^*$ of the airfoil is incorporated in the inviscid model by using a surface transpiration concept, where a non-zero normal velocity on the original airfoil boundary is prescribed. The boundary condition for the inviscid model is

$$\rho_0 v_0 = \frac{\partial}{\partial x}(\rho_e u_e \delta^*),$$

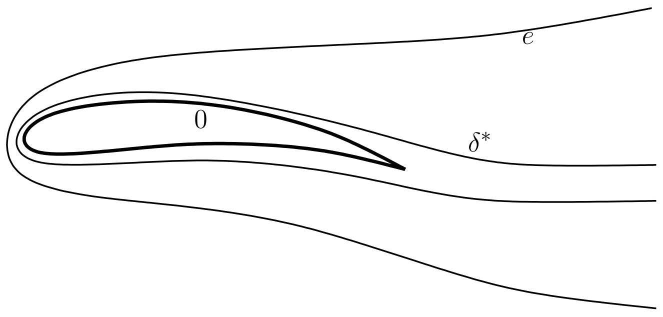


**Figure 2.1.:** Viscous–Inviscid interface $e$ and displacement thickness $\delta^*$, both exaggerated in thickness

with $\rho_0 v_0$ the mass flux normal to the airfoil and $\rho_e u_e$ the stream-wise mass flux at the edge of the viscous layer.

For the purpose of this research the exact equations used to describe the viscous and inviscid physics of the airflow are not important. It suffices to say that for a given airfoil geometry and upstream flow conditions, both models can be expressed as a relation between the coupled variables, $u_e$ and $\delta^*$.

### 2.3.1. The iterative procedure and interaction law

In the work of Veldman the viscous model and the external inviscid flow are defined as $u_e = V(\delta^*)$ and $u_e = E(\delta^*)$. Here the notation of Equation (2.2) will be used, in order to see the connection with the general formulation of an interaction law. A straightforward iterative procedure could be written as

$$\begin{aligned} \mathcal{V}(x^n) &= 0 \\ \mathcal{V}^l_\Gamma(x^n) &= \mathcal{E}^r_\Gamma(y^{n-1}) = u_e^{n-1} \\ \mathcal{E}(y^n) &= 0 \\ \mathcal{E}^l_\Gamma(y^n) &= \mathcal{V}^r_\Gamma(x^n) = \delta^{*n}. \end{aligned} \tag{2.9}$$

Here $\mathcal{V}(x) = u_e - V(\delta^*)$ and $\mathcal{E}(x) = u_e - E(\delta^*)$. The internal states $x$ and $y$ both contain $u_e$ and $\delta^*$ along the airfoil, in some not further specified way.

This procedure produces correct results for attached flows, but according to [6] it breaks down when flow separation occurs. Near the singular points of flow separation there is a strong interaction between the viscous and inviscid flow and no hierarchy between the models is present. The viscous model used for $\mathcal{V}$ has a minimum for $u_e$, corresponding to the onset of flow separation. The iterative procedure breaks down when $u_e$ below this minimum is prescribed for $\mathcal{V}$. Reversing the order of information exchange solves this problem, as $\mathcal{V}$ can be solved for any $\delta^*$, but leads to a very slow converging procedure.

An interaction law is used to obtain a stable procedure, while still solving the original combined system with as little extra computational work as possible. The interaction law consists of a simple model as an approximation of $\mathcal{E}$. This simple model is solved simultaneously with the boundary-layer equations of $\mathcal{V}$. The effect of this is that the boundary condition for $\mathcal{V}$ is not just $u_e$ anymore, but also contains the simple model. This results in a stable procedure, even in the presence of flow separation.

Usually the simple model is based on some physical representation that is simpler than that of $\mathcal{E}$. But this can lead to interaction laws that are still quite complex and increase the computational work required to solve $\mathcal{V}$ together with the simple model substantially. Because the interaction law does not influence the solution resulting from the iterative procedure, it is not strictly necessary that the simple model is derived from a physical representation of the system it describes. In order to keep the interaction law simple, the proposed simple model $\mathcal{I} : u_e = I(\delta^*)$ is constructed in a straightforward way based on the matrices describing the numerical models.

Let $\mathbf{V}$, $\mathbf{E}$ and $\mathbf{I}$ be the matrix representation of the $V$, $E$ and $I$ respectively. In the notation of Equation (2.9), $\mathbf{V}$ and $\mathbf{E}$ are the matrix representation of $[\mathbf{0}|\mathbf{Id}]^T \left[\mathcal{V}|\mathcal{V}_\Gamma^l\right] (\mathcal{V}_\Gamma^r)^{-1}$ and $\mathcal{E}_\Gamma^r \left[\mathcal{E}|\mathcal{E}_\Gamma^l\right]^{-1} [\mathbf{0}|\mathbf{Id}]$, with $\mathbf{Id}$ the identity matrix and $|$ the matrix stacking operator.

The simple model is constructed by dropping off-diagonals of $\mathbf{E}$. This means that $\mathbf{I}$ can vary from very simple, a matrix containing only the main diagonal of $\mathbf{E}$, to a complete copy of $\mathbf{E}$. The proposed revised iterative procedure as formulated in the original paper is

$$(\mathbf{I} - \mathbf{V})\delta^{*n} = (\mathbf{I} - \mathbf{E})\delta^{*n-1}. \tag{2.10}$$

Formulated as an interaction law the similarity to (2.6) is clear.

$$\tilde{\mathcal{E}}_\Gamma^r = u_e^{n-1} + \left(\mathcal{I}(\delta^{*n}) - \mathcal{I}(\delta^{*n-1})\right)$$

In [7] it is shown that this interaction law results in a procedure that can solve the coupled system in the presence of flow separation, except for the degenerate case $\mathbf{I} = \mathbf{E}$, i.e. no off-diagonals are dropped. Even when $\mathbf{I}$ is reduced to the least possible complexity, a diagonal matrix, the interaction law results in a stable procedure and the coupled system converges.

### 2.3.2. Analysis of the interaction law

To see why the interaction law results in a stable procedure, some matrix theory is required. Statements here are based on theorems developed in [8] and [9]. Exact references and more details can be found in [6] and [7].

First, $\mathbf{E} - \mathbf{V}$ and $\mathbf{I} - \mathbf{V}$ are assumed to be diagonally dominant and have positive diagonal entries and non-positive off-diagonals, with all the eigenvalues in the stable half-plane. These conditions on $\mathbf{E} - \mathbf{V}$ lead to the fact that $\mathbf{E} - \mathbf{V}$ is non-singular and $(\mathbf{E} - \mathbf{V})^{-1} \geq 0$. The iterative procedure (2.10) is the repeated application of the operator $(\mathbf{I} - \mathbf{V})^{-1}(\mathbf{I} - \mathbf{E})$. The maximum absolute value of the eigenvalues, the spectral radius $\rho$, of this operator is strictly less than unity.

$$\rho\left((\mathbf{I} - \mathbf{V})^{-1}(\mathbf{I} - \mathbf{E})\right) < 1$$

This means that the iterative procedure (2.10) is convergent and that it converges to $(\mathbf{E} - \mathbf{V})\delta^* = 0$.

Moreover, if two matrices $\mathbf{I_a}$ and $\mathbf{I_b}$ are both simplifications of $\mathbf{E}$, where $\mathbf{I_b}$ has more off-diagonals dropped or than $\mathbf{I_a}$, that is $\mathbf{I_a} \leq \mathbf{I_b}$, then the spectral radius of the iterative operator of $\mathbf{I_a}$ is not greater than that of $\mathbf{I_b}$.

$$\rho\left((\mathbf{I_a} - \mathbf{V})^{-1}(\mathbf{I_a} - \mathbf{E})\right) \leq \rho\left((\mathbf{I_b} - \mathbf{V})^{-1}(\mathbf{I_b} - \mathbf{E})\right).$$

This means that the number of iterations required to solve (2.10) increases monotonically with the number of off-diagonals dropped from $\mathbf{E}$.

Every iteration requires solving (2.10). The inverse of $\mathbf{I} - \mathbf{V}$ is not calculated directly, but the system is solved iteratively. Under the same assumptions as above it can be shown that the number of these sub-iterations decrease monotonically in the number of dropped off-diagonals in $\mathbf{I}$.

The two iterative processes show opposite convergence behaviour when the interaction law $\mathbf{I}$ is simplified. To find the method with the overall lowest number of iterations, numerical experiments have been performed by Veldman. In [7] it is shown that for both the simplest interaction law, where only the main diagonal remains, and for the case $\mathbf{I} \to \mathbf{E}$, the total number of sub-iterations required reaches a local minimum.

For the original problem to converge to a solution, all eigenvalues of $\mathbf{E} - \mathbf{V}$ should lie in the stable half-plane. As a measure of robustness, let $\tau(\ .\ )$ denote the minimum real part of the eigenvalues of a matrix. Then under the earlier assumptions for $\mathbf{E} - \mathbf{V}$ and $\mathbf{I} - \mathbf{V}$ it can be shown that

$$\tau(\mathbf{E} - \mathbf{V}) \leq \tau(\mathbf{I_a} - \mathbf{V}) \leq \tau(\mathbf{I_b} - \mathbf{V}),$$

with $\mathbf{I_a} \leq \mathbf{I_b}$. So robustness of the interaction law increases as more off-diagonals are dropped.

An outline of the proof follows. By construction we have

$$\mathbf{E} - \mathbf{V} \leq \mathbf{I_a} - \mathbf{V} \leq \mathbf{I_b} - \mathbf{V}.$$

Now if each of these these three matrices are decomposed as $\mathbf{A} = \alpha\mathbf{Id} - \hat{\mathbf{A}}$, with $\alpha$ sufficiently large such that $\hat{\mathbf{A}} \geq 0$, then the Perron–Frobenius theorem can be used. This theorem states that if $\hat{\mathbf{A}} \geq 0$ is an irreducible matrix then $\hat{\mathbf{A}}$ has a positive real eigenvalue equal to its spectral radius $\rho(\hat{\mathbf{A}})$ and $\rho(\hat{\mathbf{A}})$ increases when any entry of $\hat{\mathbf{A}}$ increases.[9]

So if $\hat{\lambda}$ is the largest real eigenvalue of $\hat{\mathbf{A}}$, then $\tau(\mathbf{A}) = \alpha - \hat{\lambda}$. Moreover $\tau(\mathbf{A})$ is non-decreasing in $\mathbf{A}$, which concludes the proof.

## 2.4. Loose coupling in rotorcraft fluid–structure interaction

Current research on the simulation of steady forward rotorcraft flight is generally based on a loose coupling method between a rotorcraft flight dynamics model and an aerodynamics solver using, for example, the Euler [10] or Reynolds-averaged Navier–Stokes [11] equations. The flight dynamics model usually consists of a CSD model describing the structure of the rotor blades, integrated with a simplified CFD model for the aerodynamics and a trim procedure. The simplified aerodynamics model can be a lookup table of experimentally found aerodynamic forces or a lifting line model, possibly extended by including unsteady effects and a vortex wake. The simplified model is usually called the 2D-CFD model, as it does not compute a full 3-dimensional flow solution. The trim procedure adjusts the angles of the flight controls, which prescribe the motion of the rotor blade at the root. The goal of the trim procedure is to adjust the control angles such that the resulting mean total rotor hub loads are appropriate for the chosen flight condition.

In order to make the results of the flight dynamics model more accurate, it is coupled to a first principles CFD model. The loose coupling procedure is outlined below and also shown in Figure 2.2.

- The flight dynamics code computes a trimmed periodic solution of a full rotor revolution. This results in blade deformation $x$ and aerodynamic forces (calculated by the simplified model) on the rotor blade $F^{\mathrm{2D}}$.

- The CFD grid is deformed according to $x$ and a flow solution is calculated. This results in more accurate aerodynamic forces $F^{3\mathrm{D}}$.
- The difference between the two aerodynamic loads is calculated, $\Delta F = F^{3\mathrm{D}} - F^{2\mathrm{D}}$. This delta load can be seen as the error of using the 2D model in the calculation of the flight dynamics.
- The rotor is trimmed again by the flight dynamics model. The total aerodynamic loads used in the trimming process is the sum of the forces calculated by the simplified model and $\Delta F$ from the last CFD solution.
- The previous three steps are repeated until convergence is achieved.

The system is said to be converged if the total loads and pitch angles do not significantly change from one coupling iteration to the next. In general if the 2D-CFD model is good enough, the loose coupling will converge without the need for relaxation.

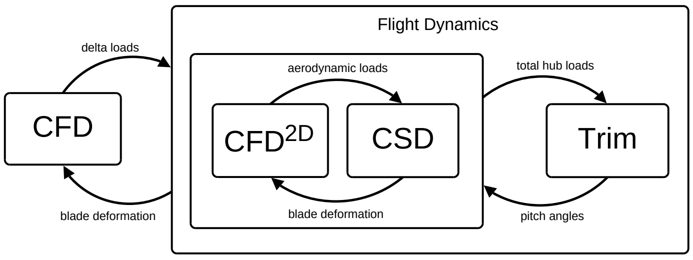


**Figure 2.2.:** Schematic overview of the loose coupling approach

### 2.4.1. Loose coupling as an interaction law

To see how this loose coupling approach fits in the framework laid out in Section 2.1, the procedure described above is presented in the formulation of Equation (2.2). First the three models and their relations are stated on a conceptual level. Then an iterative procedure for solving this combined system is proposed and finally it is shown that applying an interaction law permits the resulting procedure to be transformed into the loose coupling approach.

The basic coupled system, without using a 2D-CFD model, consists of three parts: a CSD and CFD model and a trim procedure. These three parts and their relations can be described as

$$\begin{aligned} \mathcal{S}(x) &= 0 \\ \mathcal{F}(y) &= 0 \\ \mathcal{T}(L,\theta) &= 0 \end{aligned} \qquad \begin{aligned} \mathcal{S}_{\Gamma_1}(x) &= \mathcal{F}_{\Gamma_1}(y) \\ \mathcal{S}_{\Gamma_2}(x) &= \mathcal{T}_{\Gamma_2}(L,\theta). \end{aligned} \tag{2.11}$$

Here $\mathcal{S}$ and $\mathcal{F}$ are the CSD and (full) CFD models, with the blade deflections $x$ and fluid flow state $y$. $\mathcal{T}$ is the trim procedure. For clarity its variables $L$, the mean total rotor hub loads, and $\theta$, the control angles, are written separately. It is important to note that $x$ and $y$ represent the blade deformation and flow state for every time step in an entire rotor revolution of the periodic solution. The combined system is to be solved for the free variable $\theta$ with prescribed $L$.

A possible iterative procedure to solve this system is shown in Figure 2.3. This is roughly equivalent to a tightly coupled procedure for steady trimmed rotorcraft flight. Introducing a

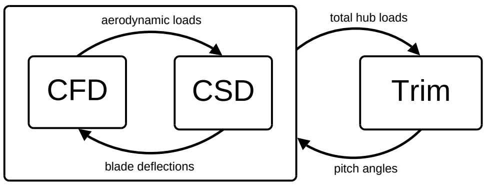


**Figure 2.3.:** Schematic overview of a basic iterative approach

superscript $n$ for the iterations in the inner loop and superscript $k$ for the outer iterations, the iterative procedure is

$$\begin{cases} \begin{cases} \mathcal{S}(x^n) = 0 \\ \mathcal{S}^l_{\Gamma_1}(x^n) = F^{3\text{D}}(y^{n-1}) \\ \mathcal{S}^l_{\Gamma_2}(x^n) = \theta^{k-1} \end{cases} \\ \begin{cases} \mathcal{F}(y^n) = 0 \\ \mathcal{F}^l_{\Gamma_1}(y^n) = x^n \end{cases} \end{cases} \\ \begin{cases} \mathcal{T}(L^k, \theta^k) = 0 \\ \mathcal{T}^l_{\Gamma_2}(L^k, \theta^k) = L(x^k). \end{cases} \tag{2.12}$$

First, the CSD and CFD models are iteratively solved with a fixed trim angle $\theta^{k-1}$. The boundary condition of $\mathcal{S}$ consists of the aerodynamic forces $F^{3\text{D}}$ and the boundary condition of $\mathcal{F}$ are the blade deflections $x$. When this combined procedure has converged the solution of the CSD model $x^k$ is used to calculate the mean rotor hub loads $L$. Using the difference between this $L$ and the target loads, the trim procedure $\mathcal{T}$ calculates the new $\theta^k$ and the next outer iteration can begin. In order for the trim procedure to be able to calculate $\theta^k$, it must have some knowledge of the effect of adjustments to $\theta$ on $L$. This can be for example an (estimated) influence matrix $\frac{\partial L}{\partial \theta}$.

### Adding the simplified aerodynamics

In order to speed up the convergence of the coupled system, an interaction law is introduced. The interaction law used instead of $F^{3\text{D}}$ is very similar to the one shown in Equation (2.6). $\mathcal{S}$ will be solved with $\tilde{F}^{3\text{D}}$, an approximation for $F^{3\text{D}}(y^n)$, instead of $F^{3\text{D}}(y^{n-1})$. In the interaction law $F^{2\text{D}}$, the aerodynamic forces on the rotor blade as calculated by the simplified model, is used to estimate the difference between $F^{3\text{D}}(y^{n-1})$ and $F^{3\text{D}}(y^n)$.

$$\tilde{F}^{3\text{D}} = F^{3\text{D}}(y^{n-1}) + \left(F^{2\text{D}}(x^n) - F^{2\text{D}}(x^{n-1})\right)$$

The goal of the interaction law is to allow the combined system to be solved with fewer iterations of the (computationally intensive) CFD model. This is done by moving the $\mathcal{F}$ from the inner to the outer loop. To make that possible, first the terms of $\tilde{F}^{3\text{D}}$ are reordered, such that a delta load as used in the loose coupling procedure appears.

$$\begin{aligned} \tilde{F}^{3\text{D}} &= F^{2\text{D}}(x^n) + \left(F^{3\text{D}}(y^{n-1}) - F^{2\text{D}}(x^{n-1})\right) \\ &= F^{2\text{D}}(x^n) + \Delta F^{n-1} \end{aligned}$$

Now instead of updating $\Delta F^n$ in every iteration of $\mathcal{S}$, a fixed $\Delta F$ is used. The result of this is that the CSD and CFD models are detached. The boundary condition $\tilde{F}^{3\text{D}}$ and thus the solution procedure for $\mathcal{S}$ does not depend anymore on information from $\mathcal{F}$ at inner iteration $n-1$. This allows for $\mathcal{F}$ and $\mathcal{T}$ to swap places in the iterative procedure, such that the $\mathcal{S}$–$\mathcal{T}$ iterations are now the inner loop and the $(\mathcal{S}\mathcal{T})$–$\mathcal{F}$ iterations are the outer loop. The revised iterative procedure using the interaction law is then described by

$$\begin{cases} \begin{cases} \begin{cases} \mathcal{S}(x^n) = 0 \\ \mathcal{S}^l_{\Gamma_1}(x^n) = F^{2\mathrm{D}}(x^n) + \Delta F^{k-1} \\ \mathcal{S}^l_{\Gamma_2}(x^n) = \theta^{n-1} \end{cases} \\ \begin{cases} \mathcal{T}(L^n, \theta^n) = 0 \\ \mathcal{T}^l_{\Gamma_2}(L^n, \theta^n) = L(x^n) \end{cases} \end{cases} \\ \begin{cases} \mathcal{F}(y^k) = 0 \\ \mathcal{F}^l_{\Gamma_1}(y^k) = x^k \end{cases} \\ \text{with } \Delta F^k = F^{3\mathrm{D}}(y^k) - F^{2\mathrm{D}}(x^k). \end{cases} \tag{2.13}$$

This procedure to solve the original system (2.11) is exactly the formalised version of the described loose coupling approach.

### 2.4.2. Remarks on the loose coupling approach

#### The role of the interaction law

It is important to note that the primary goal of the interaction law for the loose coupling approach is different than for the examples described in Section 2.2 and 2.3. These previous examples involved only two coupled models and the interaction law was necessary to facilitate convergence of the coupling iterations. In contrast, the case described here involves three models that have to be coupled. Also, in general the stability of the $\mathcal{S}$–$\mathcal{F}$ coupling is not a problem for practical rotorcraft aeroelastic simulations. The main problem is the computational load required for solving the CFD model to the required accuracy without a prescribed trim.

While the interaction law will also have a positive influence on the convergence of the coupled system, its main purpose here is to enable the CFD model to be moved to the outer loop. This reduces the number of times this computationally expensive model needs to be solved.

#### Hierarchy in model coupling

The iterative treatment of the three coupled models has been shown with two different hierarchies. The first with $\mathcal{S}$–$\mathcal{F}$ in the inner loop and the second with $\mathcal{S}$–$\mathcal{T}$ in the inner and $\mathcal{F}$ in the outer loop. This suggests the possibility of a third method, in which the inner loop consists of $\mathcal{F}$–$\mathcal{T}$ and the CSD model is solved in the outer loop.

Because $\mathcal{F}$ and $\mathcal{T}$ are not directly coupled, but only indirectly through $\mathcal{S}$, there is nothing to iterate in the inner $\mathcal{F}$–$\mathcal{T}$ loop. Thus this approach is actually the same as a non-hierarchical approach, in which the three models are solved consecutively in one iterative loop. The independence of $\mathcal{F}$ and $\mathcal{T}$ in the inner loop could be exploited by solving the two models at the same time, in parallel. However for rotorcraft simulation this has no practical advantages, because solving $\mathcal{T}$ is trivial compared to solving $\mathcal{F}$.

The model hierarchy with an $\mathcal{S}$–$\mathcal{F}$ inner loop could also be restored using an approximation of $\mathcal{S}$ as an interaction law. It is deemed unlikely however that an interaction law can be found that is simple but preserves enough of the behaviour of $\mathcal{S}$ to be useful. More importantly, with this approach $\mathcal{F}$ still resides in the inner loop of the hierarchy, with the corresponding computational disadvantages, as discussed earlier.

### Convergence and consistency of the modified procedure

Provided that the problem of solving the combined model is well posed, the choice of model for the simplified aerodynamics does not influence the solution of the converged system. For if the loose coupling converges then $x^n = x^{n-1}$ and thus $F^{2\mathrm{D}}(x^n) = F^{2\mathrm{D}}(x^{n-1})$. So in the boundary condition $\mathcal{S}^l_{\Gamma_1}$ from (2.13) the two $F^{2\mathrm{D}}$ terms cancel and only $F^{3\mathrm{D}}(y^{k-1})$ remains. This means that the loose coupling as described by (2.13) will converge to the same solution as the original procedure (2.12), irrespective of the choice of the simple model $F^{2\mathrm{D}}$. Of course it is still possible that the loose coupling is not stable at all, perhaps even when the original procedure is. In particular, it is expected that when the simple model is not a sufficiently good approximation of the CFD model, the loose coupling will fail to converge.

In [10] it was shown that for a practical rotorcraft aeroelastic coupling the choice of 2D model did have an influence on the solution of the coupled system. The solution was different when another simplified model was used and neither matched the tight coupling solution used as a reference. This was attributed to the fact that the CSD and CFD models were only coupled in a few variables along the rotor blade. If for example only lift and drag are coupled between the CFD model and the structural model and the lifting line 2D model additionally calculates a pitching moment, the solution will depend on the choice of 2D model.

Another possible reason for different solutions found by the loose coupling is that the problem of finding a trimmed solution is not well posed at all, i.e. there are multiple solutions. As the choice of interaction law does influence the convergence path, it is possible that by changing the 2D model a different solution is found. This does not have to be a problem however, because both solutions are equally valid. The solution can be checked by confirming that it is also a solution of the tightly coupled procedure.

The treatment of the model coupling in this chapter presumes that each of the models solves the entire periodic solution at once. In practice however, for example in Servera[10], but also in the model problem described in Chapter 4, the coupling between the CSD model and either the full or the simplified CFD model is done at every time step. This can alter the convergence path, but should not result in a different converged periodic solution.

CHAPTER 3

# Autoregressive modelling of time series

A time series can be defined as a set of quantitative observations arranged in chronological order. These observations can be analysed [12] and the properties of the time series derived from the analysis can then be used to predict future development of the series.

Although based in economic modelling, time series analysis is also a valuable tool in engineering. In aircraft design for example, to analyse wing flutter, the motion of the wing is modelled using an autoregressive model, often augmented with a moving average part.

In the context of this research, the interpretation of a time series is extended to a general series of (possibly higher dimensional) data points. The data points of the time series do not correspond to the value of some variable at a certain point in time, but to the intermediate result after one iteration of an iterative procedure. To capture the properties of the time series, an autoregressive model is used.

## 3.1. Autoregressive model

A discrete time stochastic process $X_t$ is called an autoregressive process of order $p$, an AR($p$) process, if it can be described by the difference equation,

$$X_t = \delta + \phi_1 X_{t-1} + \ldots + \phi_p X_{t-p} + u_t, \tag{3.1}$$

where $\delta$ is a constant term and $u_t$ a stochastic process, i.e. for every $t \in \mathbb{N}$, $u_t$ is a random variable. The $p$ coefficients $\phi_j$ are the fixed parameters of the AR($p$) process.

From now on only first order autoregressive processes will be considered. If $X_t$ is assumed to be a mean zero AR(1) process, Equation (3.1) simplifies to

$$X_t = \phi X_{t-1} + u_t, \tag{3.2}$$

with $u_t$ a mean zero stochastic process.

Suppose at some time $t_0$ the value of $X_{t_0}$ is known, then $X$ at a later time $t = t_0 + \tau$ can be described by

$$X_{t_0+\tau} = \phi^\tau X_{t_0} + \sum_{j=0}^{\tau-1} \phi^j u_{t_0+\tau-j}.$$

As all the random variables $u_j$ have a zero mean, the expectation of $X_{t_0+\tau}$ is

$$E[X_{t_0+\tau}] = \phi^\tau X_{t_0}. \tag{3.3}$$

So given the stated assumptions, the future of $X$ can be predicted using an initial value $X_{t_0}$ and the single parameter $\phi$.

### 3.1.1. Parameter estimation

In general the value of the parameter $\phi$ is not known a priori, but must be estimated from the time series. A series of observations $X_0, \ldots, X_T$ can be used to calculate the $T$ values of $u_t$, according to Equation (3.2). The $u_t$ terms can be interpreted as error terms. Fitting the model by minimising the total error in a least squares sense therefore means that the best estimate for $\phi$ is the one that minimises

$$\sum_{t=1}^{T} |X_t - \phi X_{t-1}|^2 .$$

This leads to

$$\phi = \frac{\sum_{t=1}^{T} X_t X_{t-1}}{\sum_{t=0}^{T-1} X_t^2}. \tag{3.4}$$

The fitted model is stable if $|\phi| < 1$.

## 3.2. Vector Autoregressive model

In a vector autoregressive (VAR) process every observation in the time series is a vector. The formulation of Equation (3.2) still holds for a VAR(1) process, but now with $X_t$ and $u_t$ as $N \times 1$ vectors and $\phi$ replaced by $\mathbf{\Phi}$, an $N \times N$ matrix.

Again, the parameter matrix $\mathbf{\Phi}$ can be estimated by solving for the smallest errors in the observations in the least squares sense. For a series of observations $X_0, \ldots, X_T$ the estimated parameter matrix is

$$\begin{aligned} \mathbf{\Phi} &= BA^{\mathrm{T}} \left(AA^{\mathrm{T}}\right)^{-1} \\ \text{with:} \quad A &= \begin{bmatrix} X_0 & \cdots & X_{T-1} \end{bmatrix} \\ B &= \begin{bmatrix} X_1 & \cdots & X_T \end{bmatrix}. \end{aligned} \tag{3.5}$$

The fitted model is stable if every eigenvalue $\lambda$ of $\mathbf{\Phi}$ lies within the unit circle, $|\lambda| < 1$.

### 3.2.1. Subset VAR model

In a VAR(1) model, every variable in the $N \times 1$ vector $X_t$ depends on every variable of the previous observation $X_{t-1}$. In other words, the matrix $\mathbf{\Phi}$ is a full matrix. This does not necessarily reflect the causal relationships between the variables of the process that is being modelled. Moreover, it can be shown that using a model with more degrees of freedom in the parameter matrix than there are in the underlying process results in less accurate predictions. For more information see the detailed explanations by Lütkepohl[13].

A subset VAR model uses a sparse matrix to limit the number of autoregressive coefficients in $\mathbf{\Phi}$. Apart from a more accurate modelling of the autoregressive process, a subset VAR model is also useful when the size $N$ of the vector $X_t$ is larger than the available number of observations $T$. In this case the choice of the full parameter matrix $\mathbf{\Phi}$ is not uniquely defined. The parameter matrix can be uniquely defined from the observations $X_0, \ldots, X_T$ if only up to $TN$ non zero elements are allowed.

In general there will be no a priori knowledge about the relationships between the variables of the autoregressive process. This means that the decision of whether to include an autoregression parameter in the model or to restrict the corresponding element of $\mathbf{\Phi}$ to zero has to be made based on the available observations. Outlined below is a procedure as described in [13].

A good model is one that achieves a low total error with respect to the observed time series while using as few nonzero parameters in $\mathbf{\Phi}$ as possible. To formalise this notion of a good subset model, the Akaikes Information Criterion (AIC) is used.

$$\begin{aligned} \text{AIC} &= \ln \tilde{\sigma}^2 + \frac{2}{T} n_{\text{param}} \\ \tilde{\sigma}^2 &= \frac{1}{T} \sum_{t=1}^{T} \left| X_t - \mathbf{\Phi} X_{t-1} \right|^2 , \end{aligned} \tag{3.6}$$

with $n_{\text{param}}$ the number of nonzero elements of $\mathbf{\Phi}$. The squared norm $|.|^2$ is the sum of the squares of the matrix elements, so $\tilde{\sigma}^2$ denotes the prediction error averaged over all the observations. The subset model chosen is the one that minimises the AIC. This strategy balances both requirements of a good model.

Unfortunately, even for medium sized problems it is unfeasible to check all the possible restrictions, because there are $2^{N^2}$ possible combinations. Instead a top-down strategy is used where, starting from a full VAR(1) model, coefficients are restricted to zero one at a time. If the restricted model yields a lower AIC that restriction is kept, otherwise the parameter is included in $\mathbf{\Phi}$ again. Because Equation (3.5) can be split in $N$ separate equations, the parameter estimation given a specific restriction can be done independently per row. To make computations easier, the calculation of the AIC is also done per row.

The procedure to find the optimal restricted parameter matrix is as follows. The first row of the matrix $\mathbf{\Phi}$ is estimated using the least squares method and the corresponding AIC is evaluated. Then the first parameter is restricted to zero and the remaining coefficients are estimated again. If the new AIC is lower, then the first parameter kept restricted, otherwise it is again included in the model. The same procedure is repeated for the second parameter and so on, until the complete set of restrictions for the first row of $\mathbf{\Phi}$ is obtained. The same is done for the other rows of the parameter matrix. The resulting matrix has nonzero entries for the parameters that most accurately describe the modelled process.

CHAPTER 4

# Model problem

The model problem consists of a closed one-dimensional cylinder filled with gas, coupled to a spring–loaded piston. See Figure 4.1 for a schematic overview. The attachment point of the spring follows a prescribed, periodic motion. This model is similar to the one used in [1], but with the spring attached to a moving wall. The goal of this model problem is to mimic the qualitative behaviour of the CFD, CSD and trim procedures of a full rotorcraft simulation with a model that is as simple as possible. This model can then be used to analyse different coupling strategies.

In the model problem the mass–spring system represents the CSD model for the rotor blades. The CFD model of the gas in the piston delivers a non-trivial force on the piston, analogous to how the complex aerodynamic forces deform the rotor blades in rotorcraft. The forced periodic motion of the spring attachment point induces a periodic solution of the combined fluid-structure system. This represents the periodicity of steady forward flight conditions. The trimming process is modelled by adjusting the spring constant. This indirectly alters the motion of the piston and the aerodynamic forces exerted on it.

The model equations are presented in a space–time formulation and subsequently discretised with a finite volume approach. This is roughly based on the discontinuous Galerkin finite element method of Van der Vegt and Van der Ven[14]. An important advantage of this approach for rotorcraft modelling is that local mesh refinement can be done naturally for both the space and time dimensions, which is useful for capturing blade vortices. Furthermore, the periodicity requirement becomes just another boundary condition, which is useful for modelling rotorcraft in stationary flight. For the model problem, however, both advantages will not be used, in order to keep the numerical procedure simple.

## 4.1. Governing Equations

The fluid in the cylinder of length $l(t)$ is described by the Euler equations of gas dynamics in conservative form, which in space–time formulation can be written as:

$$\mathcal{F} : \quad \nabla \cdot F(U) = 0 \tag{4.1}$$

$$\text{with} \quad F = \begin{bmatrix} \rho u & \rho \\ \rho u^2 + p & \rho u \\ \rho u e + p u & \rho e \end{bmatrix} \quad , \quad U(\bar{x}) = \begin{bmatrix} \rho \\ \rho u \\ \rho e \end{bmatrix} \quad , \quad \bar{x} = \begin{bmatrix} x \\ t \end{bmatrix}$$

where

| | |
|---|---|
| $\rho$ : | mass density |
| $u$ : | fluid velocity |
| $e$ : | total energy |
| $p$ : | pressure |
| $x$ : | length along the cylinder |
| $t$ : | time |

This system is closed by using the state equation for a perfect gas

$$p = (\gamma - 1)(\rho e - \frac{1}{2}\rho u^2),$$

with the ratio of specific heats $\gamma$ assumed to be 1.4.

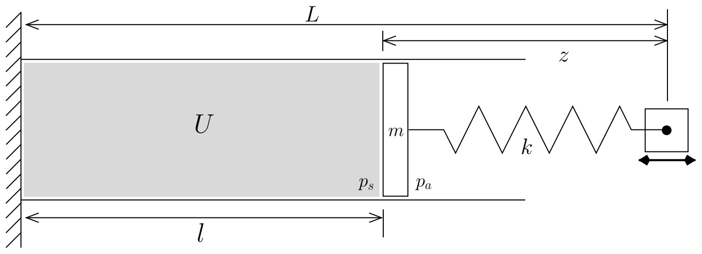


**Figure 4.1.:** Piston model problem

The structure is modelled with an ordinary differential equation describing a mass–spring system with one degree of freedom:

$$\mathcal{S}: \quad m(\ddot{L} + \ddot{z}) + kz = (p_s - p_a)A. \tag{4.2}$$

Here $z$ is the displacement of the structure relative to the spring equilibrium position, in the same direction as $L$, so a positive $\dot{z}$ means an increasing cylinder volume. $m$ is the mass of the piston, $k$ the spring constant and $A$ the surface area of the piston. The forcing term of the spring equation is the difference of the force exerted by the fluid on the piston $p_sA$ and the force resulting from the atmospheric pressure $p_aA$ outside the cylinder.

To describe the motion of the wall the spring is attached to, the equilibrium position is a prescribed function of time $L(t)$. The motion is chosen to be a simple sinusoidal with period $t_L$, mean value $\bar{L}$ and amplitude $\Delta L$.

$$L(t) = \bar{L} + \Delta L \cos(\omega_L t),$$
$$\text{with} \quad \omega_L = \frac{2\pi}{t_L}$$

### 4.1.1. Boundary conditions and model coupling

At the fluid–structure interface conservation of mass, momentum and energy is satisfied by the following dynamic and kinematic conditions.

$$\mathcal{F}_\Gamma \equiv \begin{bmatrix} p_\Gamma \\ l \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} p_s \\ L + z \\ \dot{L} + \dot{z} \end{bmatrix} \equiv \mathcal{S}_\Gamma \tag{4.3}$$

The variables $p_\Gamma$ and $u_\Gamma$ denote the fluid pressure and velocity at the fluid–structure boundary. Together (4.1), (4.2) and (4.3) describe the coupled fluid–structure system.

Due to the forced motion of the spring attachment point, work is done by the piston on the gas. This work results in a net increase of energy in the fluid system. In order to obtain a periodic solution the total internal energy of the system must be bounded. To this end the left wall is kept at a fixed temperature. The moving right wall is modelled as an adiabatic wall, as this results in a simpler boundary condition than the isothermal wall. The idea of this setup is that energy added to the gas by the work of the piston can leave the system at the isothermal left wall as an outward heat flux.

For $\mathcal{S}$ initial conditions $z_I$ and $\dot{z}_I$ are given. $\mathcal{F}$ can either have an initial condition or a periodicity requirement. For the model problem an initial condition is prescribed by the fluid state that is defined by the parameters $\rho_I$, $u_I$, $p_I$.

### 4.1.2. Nondimensionalisation

All the variables and equations are nondimenionalised. The reference quantities for the scaling are derived from the length $\bar{L}$, density $\rho_I$ and pressure $p_I$. Additionally, the fluid variables are scaled with the cylinder cross-section area $A$, such that Equation (4.1) describes the fluid system in one-dimensional form. No new notation will be introduced, but instead all variables are assumed nondimensional from now on. All relevant variables are listed in Table A.2 and the dimensional values of the parameters are listed in Table A.1.

## 4.2. Fluid discretisation

The fluid is discretised using a space–time finite volume approach. In the time direction the space-time domain is partitioned in evenly spaced space–time slabs of height $\Delta t$. The domain $\Omega$ for the space–time slab $t \in [t_0, t_1]$ is depicted in Figure 4.2. The motion of the right boundary is linearly interpolated between the values $l_0 = l(t_0)$ and $l_1 = l(t_1)$.

Along the spatial direction, $\Omega$ is divided into $N$ evenly distributed elements. The length of an element in the $x$-direction is $h_0 = l_0/N$ and $h_1 = l_1/N$ at time $t_0$ and $t_1$ respectively. The element boundary $\partial\Omega_i$ is divided into four parts. $S_-$ and $S_+$ are the time boundaries of the element, at time $t_0$ and $t_1$ respectively and $S_{i-1}$ and $S_i$ are the left and right boundary, internal to the space–time slab.

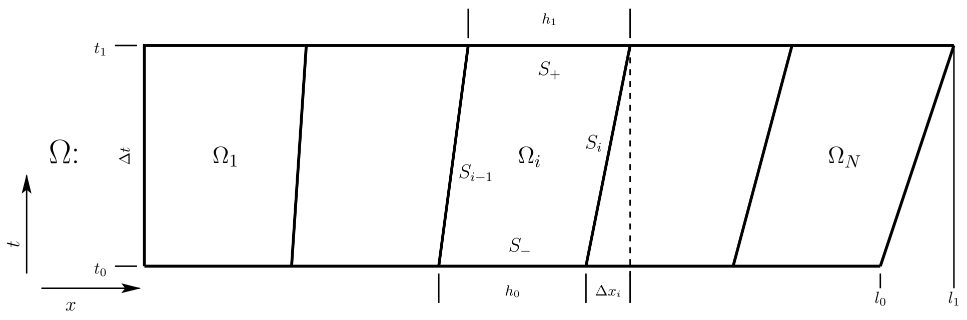


**Figure 4.2.:** Fluid discretisation in one space–time slab

In each element $\Omega_i$ the flow state is assumed to have a constant value, denoted $U_{\Omega_i}$.

To construct the finite volume discretisation Equation (4.1) is integrated over an element $\Omega_i$ and the divergence theorem is applied. This results in

$$\begin{aligned} 0 &= \int_{\Omega_i} \nabla \cdot F(U) d\Omega \\ &= \int_{\partial\Omega_i} F(U) \cdot \bar{n} \; dS \end{aligned}$$

with $\bar{n}$ the unit outward space–time normal vector of the element boundary.

The length of each part of the boundary and the unit normal vector pointing in the positive $t$ direction for $S_-, S_+$ and in the positive $x$ direction for $S_i$ are given by

$$\begin{aligned} \|S_-\| = h_0 \qquad\qquad \|S_+\| = h_1 \qquad\qquad \|S_i\| = \sqrt{\Delta t^2 + \Delta x_i^2} \\ \bar{n}_- = \bar{n}_+ = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad\qquad \bar{n}_i = \begin{bmatrix} \Delta t \\ -\Delta x_i \end{bmatrix} \frac{1}{\|S_i\|} \quad . \end{aligned}$$

The integral over the element boundary $\partial\Omega_i$ can now be split into four parts. If $F(U)$ is assumed to have a unique value $F_S$, constant on each of element boundary parts, the surface integral can be reduced to

$$\begin{aligned} \int_{\partial\Omega_i} F(U) \cdot \bar{n} \; dS &= \int_{S_+} F(U) \cdot \bar{n} dS + \int_{S_-} F(U) \cdot \bar{n} dS + \int_{S_i} F(U) \cdot \bar{n} dS + \int_{S_{i-1}} F(U) \cdot \bar{n} dS \\ &= \int_{S_+} F_{S_+} \cdot \bar{n}_+ dS - \int_{S_-} F_{S_-} \cdot \bar{n}_- dS + \int_{S_i} F_{S_i} \cdot \bar{n}_i dS - \int_{S_{i-1}} F_{S_{i-1}} \cdot \bar{n}_{i-1} dS \\ &= F_{S_+} \cdot \bar{n}_+ \|S_+\| - F_{S_-} \cdot \bar{n}_- \|S_-\| + F_{S_i} \cdot \bar{n}_i \|S_i\| - F_{S_{i-1}} \cdot \bar{n}_{i-1} \|S_{i-1}\|. \end{aligned} \quad (4.4)$$

## 4.2.1. Numerical flux

As $U$ is not uniquely defined at the element boundary $S_j$ the flux $F(U)$ can not directly be used in (4.4) as the value of $F_S$. Instead a numerical flux is introduced. For the purpose of this model problem, with the flow state assumed constant in an element, a simple Rusanov flux, as described by Toro[15], will suffice. This flux is used at every part of the element boundary. The left state $U_L$ is the state inside the element and the right state $U_R$ the state in the adjacent element.

$$\begin{aligned} F_S \cdot \bar{n} &\approxeq H(U_L, U_R, \bar{n}) \\ &= \frac{1}{2} \left( F(U_L) + F(U_R) \right) \cdot \bar{n} + \frac{1}{2} (U_L - U_R) \bar{\lambda}, \end{aligned}$$

$$\text{with} \qquad \bar{\lambda} = \max(\bar{\lambda}_L, \bar{\lambda}_R) \qquad \bar{\lambda}_{L/R} = \max \left| \begin{bmatrix} u & 1 \\ u - a & 1 \\ u + a & 1 \end{bmatrix} \cdot \bar{n} \right| \qquad a = \sqrt{\gamma p / \rho}$$

The numerical flux is conservative, i.e. $H(U_L, U_R, \bar{n}) = -H(U_R, U_L, -\bar{n})$, so it can be used in (4.4).

At the element boundaries between the different space–time slabs, $S_-$ and $S_+$, this numerical flux yields

$$F_{S_-} \cdot \bar{n} = U_{\Omega_i}^{t_0} \quad , \qquad F_{S_+} \cdot \bar{n} = U_{\Omega_i},$$

with $U_{\Omega_i}^{t_0}$ the value of $U$ at element $\Omega_i$ in the previous space–time slab.

At the interior boundary parts in the space–time slab, $S_i$, the flux is

$$F_{S_i} \cdot \bar{n} = H_i \qquad \text{with} \qquad H_i = H(U_{\Omega_i}, U_{\Omega_{i+1}}, \bar{n}) \text{ for } i = 1, \ldots, N-1.$$

For $i = 0, N$ the flux $H_i$ must come from the boundary conditions.

Applying the expressions for $F_S \cdot \bar{n}$ in (4.4) and rewriting them using $\hat{H}_i = H_i \|S_i\| \Delta t^{-1}$ and $\hat{\lambda} = \bar{\lambda} \|S_i\| \Delta t^{-1}$ yields the following equation for each element $\Omega_i$.

$$h_1 U_{\Omega_i} - h_0 U_{\Omega_i}^{t_0} + \Delta t \left( \hat{H}_i - \hat{H}_{i-1} \right) = 0 \tag{4.5}$$

$$\text{with} \quad \hat{H}_i = \frac{1}{2} \left( F(U_L) + F(U_R) \right) \cdot \begin{bmatrix} 1 \\ -v \end{bmatrix} + \frac{1}{2} (U_L - U_R) \hat{\lambda},$$

$$\hat{\lambda} = \max \begin{pmatrix} |u_L - v| + a_L \\ |u_R - v| + a_R \end{pmatrix}, \qquad v = \frac{\Delta x_i}{\Delta t}$$

Here the wave speed $\hat{\lambda}$ used in the approximate solution of the Riemann problem is an upper bound of the characteristic wave speeds, as described in [15]. Note how the length of the boundary cancels with the scaling of the normal and how the inner product with $\bar{n}$ in the equation for $\bar{\lambda}$ results in the correction of the fluid velocity with the element boundary velocity.

### 4.2.2. Boundary conditions

At the left and right boundaries of $\Omega$ a solid wall is assumed. On the left side of the domain the wall is stationary and on the right side the wall is moving. Given the velocity $v = \frac{\Delta x}{\Delta t}$ of the wall and the flow state $U$ at the wall, the exact flux at the left and right boundaries of the space–time slab can be calculated

$$\begin{aligned} F(U) \cdot \bar{n} &= F(U) \cdot \begin{bmatrix} \Delta t \\ -\Delta x_i \end{bmatrix} \frac{1}{\|S_i\|} \\ &= F(U) \cdot \begin{bmatrix} 1 \\ -v \end{bmatrix} \frac{\Delta t}{\|S_i\|} \\ &= \begin{bmatrix} \rho u - \rho v \\ \rho u^2 + p - \rho u v \\ \rho u e + p u - \rho e v \end{bmatrix} \frac{\Delta t}{\|S_i\|} \\ &= \begin{bmatrix} \rho (u - v) \\ \rho u (u - v) + p \\ \rho e (u - v) + p u \end{bmatrix} \frac{\Delta t}{\|S_i\|}. \end{aligned}$$

With the requirement for the solid wall, $u = v$, the flux simplifies to

$$F(U) \cdot \bar{n} = \begin{bmatrix} 0 \\ p \\ p v \end{bmatrix} \frac{\Delta t}{\|S_i\|}. \tag{4.6}$$

This means that the only variable from the flow state that needs to be specified at the boundary is the pressure.

#### Right boundary: moving adiabatic wall

Because the length of the fluid domain is interpolated linearly between $l_0$ and $l_1$, the velocity of the moving right wall, $v_r$, is constant.

$$v_r = \frac{l_1 - l_0}{\Delta t}$$

In the case of an adiabatic wall, the pressure at the wall should be such that there is no heat transferred through the wall.

In a first attempt to achieve this, the pressure of the interior is simply extrapolated to the wall. With the notation used in describing a Riemann problem, where $p_L$ denotes the pressure in the element next to the wall and $p^*$ the pressure at the wall, this approach can be written as

$$p^* = p_L.$$

The results are however unsatisfactory. With increasing resolution of the space-time grid the particle velocity in the element adjacent to the wall does not converge to the speed of the wall.

A second approach for modelling an adiabatic moving wall is described by Toro[15]. The pressure comes from the solution of a Riemann problem where the left state is the state in the element next to the wall and the right state comes from a dummy element. If the left state is defined by $\rho_L$, $u_L$ and $p_L$, then the dummy state is given by the conditions $\rho_R = \rho_L$, $u_R = -u_L + 2v$ and $p_R = p_L$. These particular left and right states allow the Riemann problem to be solved analytically. The solution consists of either two shocks or two rarefaction waves. If local subsonic flow is assumed, the fluid state at the wall is the star state from the Riemann problem. The fluid velocity at the wall is then $u^* = v$, suggesting that the dummy state was chosen correctly to describe a wall moving with velocity $v$. The pressure at the wall is given by

$$p^* = \begin{cases} p_L \left(1 + \frac{\gamma-1}{2} \frac{u_L - v}{a_L}\right)^{\frac{2\gamma}{\gamma-1}} & \text{if } u_L \leq v, \\ p_L \left(1 + \gamma \frac{\gamma+1}{4} \left(\frac{u_L - v}{a_L}\right)^2 \left(1 + \sqrt{1 + \left(\frac{\gamma+1}{4} \frac{u_L - v}{a_L}\right)^{-2}}\right)\right) & \text{if } u_L > v. \end{cases} \tag{4.7}$$

Using this pressure in (4.6) yields a numerical procedure that follows the prescribed velocity of the wall accurately.

### Left boundary: fixed isothermal wall

Unfortunately, the flux associated with an isothermal boundary condition cannot be calculated as easily as for the case of an adiabatic wall. This is because the Riemann problem can not be solved analytically due to the fact that $p_R \neq p_L$. In [16] an alternative approach based on the approximate Riemann solver of Osher is suggested. A dummy state is constructed that, when used with the Osher solver, gives the desired properties at the wall ($u^* = 0$ and $T^* = T_\infty$). The resulting pressure at the wall to be used in (4.6) is

$$p^* = p_L \left(1 + \frac{\gamma - 1}{2} \frac{u_L}{a_L} \left(1 + \frac{u_L(a_L^2 - a_\infty^2)}{a_L a_\infty \frac{a_L + a_\infty}{\gamma - 1} + u_L \frac{a_L^2 + a_\infty^2}{2}}\right)\right)^{\frac{2\gamma}{\gamma-1}}.$$

The temperature requirement is expressed here as a prescribed speed of sound $a_\infty$. This speed of sound is related to the temperature by $a_\infty = \sqrt{\gamma R_{\text{gas}} T_\infty}$, with $R_{\text{gas}} = c_p - c_v$, the specific gas constant. Notice the similarity of this expression for $p^*$ to that of the adiabatic wall for the two rarefactions case with $v = 0$.

In testing this method, it was seen that the temperature at the wall was not reliably kept to a fixed value. Thus the approach of directly calculating the flux was abandoned for the isothermal left wall. Instead a dummy state is introduced and the numerical flux is calculated the same way as at the internal element boundaries. The dummy state $U_0$ used in the numerical flux is defined with

$$\rho_R = \rho_L \quad , \qquad \frac{u_L + u_R}{2} = 0 \quad , \qquad \frac{T_L + T_R}{2} = T_\infty .$$

Testing revealed that this implementation of the isothermal boundary condition was good in maintaining the prescribed temperature at the wall and adequate in prescribing a vanishing velocity at the fixed wall. The minor fluctuations of the fluid velocity at the fixed wall did not lead to a net mass flux through the boundary over a complete period.

#### Expressions for the boundary flux

The final boundary conditions can now be expressed in a way that they can be used in (4.5). The left isothermal boundary condition leads a dummy state $U_0$ that can be used in the numerical flux procedure.

$$\begin{aligned} \hat{H}_0 &= \hat{H}\left(U_0, U_{\Omega_1}\right) \\ \text{with} \quad U_0 &= \begin{bmatrix} \rho_1 \\ -\rho_1 u_1 \\ \rho_1 e_1 + \frac{2}{\gamma-1}\left(p_\infty \frac{\rho_1}{\rho_\infty} - p_1\right) \end{bmatrix} \end{aligned} \tag{4.8}$$

Here $\rho_1$, $u_1$, $e_1$ and $p_1$ come from $U_{\Omega_1}$, the flow state in the leftmost internal element.

The right adiabatic boundary condition leads to a direct expression of the flux:

$$\hat{H}_N = \begin{bmatrix} 0 \\ p^* \\ p^* v_r \end{bmatrix} , \tag{4.9}$$

with $p^*$ from (4.7) and $v_r$ the velocity of the right moving wall, the piston.

### 4.2.3. Iterative solution strategy

Applying Equation (4.5) to every element in the space–time slab $\Omega$ results in a system of equations. This system is to be solved for every space–time slab, analogous to time stepping in a classical finite volume approach with only a space dimension.

In order to solve the system described by (4.5), (4.8) and (4.9), a pseudo-time integration method is used. The quantity on the left hand side of (4.5) is called the residual, $R$. Finding the solution for $R(U) = 0$ then amounts to finding the stationary point of

$$\begin{aligned} \frac{\partial U}{\partial \tau} + R(U) = 0. \\ \text{with} \quad R(U) = h_1 U_{\Omega_i} - h_0 U_{\Omega_i}^{t_0} + \Delta t\left(\hat{H}_i - \hat{H}_{i-1}\right) \end{aligned}$$

The stationary point can be found with an iterative procedure.

$$\begin{aligned} U^{n+1} &= U^n - (\Delta\tau)^{-1} R(U^n) \\ \text{with} \quad \Delta\tau &= \frac{h_0 + h_1}{2} + \Delta t \max(\hat{\lambda}_i). \end{aligned} \tag{4.10}$$

Here $\Delta\tau$ is chosen such that a stable procedure results, which converges to a solution $U$ in the space–time slab $\Omega$ with a vanishing residual.

## 4.3. Structure discretisation

The pressure at the fluid–structure interface is assumed to be constant in a space–time slab, so if the forced motion of the spring $L$ is interpolated linearly on a space–time slab, Equation (4.2) can be solved analytically. The solution $z(t)$ has the general form

$$z(t) = \alpha_0 \sin(\omega(t - t_0)) + \alpha_1 \cos(\omega(t - t_0)) + \beta_0 + \beta_1(t - t_0),$$
$$\text{with} \quad \omega = \sqrt{\frac{k}{m}} \quad .$$

To satisfy the linear part of (4.2), the parameters $\beta_0$ and $\beta_1$ must be

$$\beta_0 = \frac{(p_s - p_a)A}{k} - \frac{m}{k}\ddot{L}(t_0)$$
$$\beta_1 = -\frac{m}{k}\frac{1}{\Delta t}\left(\ddot{L}(t_1) - \ddot{L}(t_0)\right).$$

The remaining parameters $\alpha_0$ and $\alpha_1$ are used to satisfy the initial conditions $z(t_0) = z_0$ and $\dot{z}(t_0) = \dot{z}_0$.

$$\alpha_0 = \frac{\dot{z}_0 - \beta_1}{\omega}$$
$$\alpha_1 = z_0 - \beta_0$$

With these parameters known the values of $z$ and $\dot{z}$ at $t = t_1$ can be calculated.

$$z_1 = z_0 \cos(\omega\Delta t) + \frac{\dot{z}_0}{\omega}\sin(\omega\Delta t) + \beta_0(1 - \cos(\omega\Delta t)) + \beta_1\left(\Delta t - \frac{1}{\omega}\sin(\omega\Delta t)\right)$$
$$\dot{z}_1 = \dot{z}_0 \cos(\omega\Delta t) - \omega z_0 \sin(\omega\Delta t) + \beta_0\omega\sin(\omega\Delta t) + \beta_1(1 - \cos(\omega\Delta t))$$

## 4.4. Coupling methods

The fluid, structure and trim procedures of the model problem will be coupled in a way similar to the coupling in the simulation of trimmed rotorcraft aeroelastics as described in Section 2.4.

### 4.4.1. Tight coupling

The approach shown in Figure 2.3 is based on a full space–time solver for both $\mathcal{F}$ and $\mathcal{S}$, i.e. the full periodic solution of either $\mathcal{F}$ or $\mathcal{S}$ is calculated before information is exchanged. In the model problem explicit time stepping is used by solving each space–time slab separately. At each time step the models are coupled and information is exchanged. This is comparable to the tightly coupled procedure in rotorcraft simulations. By calculating a combined fluid–structure solution for a number of periods, a periodic solution is obtained without a specific periodic boundary condition.

Algorithm 1 gives an overview of how the systems are coupled. The goal is to find a periodic solution for the piston motion, a periodic fluid flow state and a value for the trim parameter $T$ such that the average aerodynamic force on the piston, $\bar{F}$, is equal to some prescribed value $F_{\text{target}}$. At the end of each outer iteration the trim procedure $\mathcal{T}$ chooses a new value $T_{i+1}$ that (hopefully) results in a better $\bar{F}_{i+1}$. The stopping criterion $\epsilon$ need not be the same for the various loops. The exact value used is not of importance to the current analysis, so it can be seen as a placeholder. The exact nature of the trim procedure and which parameter of the model it uses as the trim parameter $T$ will be described in Section 4.5.

**Algorithm 1**: Tightly coupled procedure for solving the model problem

```
initialise trim parameter $T_0$ ;   // guess
repeat $i = 1, \dots$               // ($\mathcal{T}$ loop)
    $t = 0$ ;
    $z_0 = z_I$, $\dot{z}_0 = \dot{z}_I$ ;
    initialise flow state $U_0 = U_I$ ;
    repeat $j = 0, \dots$           // time stepping
        initialise $z_1^0, \dot{z}_1^0$ ;      // guess
        repeat $k = 1, \dots$       // $\mathcal{S}$–$\mathcal{F}$ loop
            $U_j^k = \mathcal{F}(U_{j-1}, z_0, z_1^k)$ ;
            calculate $p_\Gamma$ from $U_j^k$ ;
            $\left(z_1^{k+1}, \dot{z}_1^{k+1}\right) = \mathcal{S}(p_\Gamma, z_0, \dot{z}_0)$ ;
        until $|z_1^k - z_1^{k-1}| < \epsilon$;
        $t = t + \Delta t$ ;
        $(z_0, \dot{z}_0) = (z_1, \dot{z}_1)$ ;
    until $p_\Gamma$ periodic;
    calculate $\bar{F}_i$ from periodic solution $U$ ;
    adjust trim: $T_{i+1} = \mathcal{T}(T_i, \bar{F}_i)$ ;
until $|\bar{F}_i - F_{target}| < \epsilon$;
```

### 4.4.2. Loose coupling

As shown in Figure 2.2, the basis of the loose coupling approach consists of the tight coupling with the CFD model replaced by a simplified model. Once the CSD and the simplified fluid model have converged to a trimmed periodic solution, the full CFD model is used to calculate a correction on this estimate. The input for this corrector step is a vector $\mathbf{z}$ with the displacement of the piston for one period. The result of the CFD model is used to construct a vector $\mathbf{\Delta p}$ which is the difference of the periodic pressure at the piston calculated by the simplified model and the CFD model.

$$\mathbf{\Delta p} = \mathbf{p_\Gamma} - \mathbf{\hat{p}_\Gamma} \tag{4.11}$$

The variables written in a bold face denote a vector with length $N_t$, the number of time steps in one period.

$$N_t = t_L/\Delta t$$

The pressure difference $\mathbf{\Delta p}$ is then used in the next predictor step, where it is added to the pressure calculated by the simple model $\hat{p}_\Gamma$. This result is the pressure from the fluid on the pressure $p_s$.

$$\mathbf{p_s} = \mathbf{\hat{p}_\Gamma} + \mathbf{\Delta p} \tag{4.12}$$

Due to the addition $\mathbf{\Delta p}$ in the coupled procedure, the trim parameter needed to obtain $F_{\text{target}}$ will be different. However, because in this step the simple model is used, finding the new trim parameter using multiple $\mathcal{S}$–$\mathcal{F}$ loops is not computationally expensive. This process of predictor and corrector steps repeats until the average aerodynamic force as calculated by the CFD model on the piston is close enough to the target value. The complete procedure is shown in Algorithm 2.

**Algorithm 2**: Loosely coupled procedure for solving the model problem

```
initialise trim parameter $T_0$ ;   // guess
initialise $\mathbf{\Delta p}^0(j) = 0 \quad j = 0, \dots, N_t$ ;
repeat $n = 1, \dots$                // $\mathcal{F}$ loop
    repeat $i = 1, \dots$            // $\mathcal{S}$–$\mathcal{T}$ loop
        $t = 0$ ;
        $z_0 = z_I,\ \dot{z}_0 = \dot{z}_I$ ;
        repeat $j = 0, \dots$        // time stepping
            initialise $z_1^0, \dot{z}_1^0$ ;   // guess
            repeat $k = 1, \dots$    // simplified $\mathcal{S}$–$\mathcal{F}$ loop
                calculate $\hat{p}_\Gamma$ with simplified model ;
                $p_s = \hat{p}_\Gamma + \mathbf{\Delta p}^n(j \bmod N_t)$ ;
                $\left(z_1^{k+1}, \dot{z}_1^{k+1}\right) = \mathcal{S}(p_s, z_0, \dot{z}_0)$ ;
            until $|z_1^k - z_1^{k-1}| < \epsilon$;
            $t = t + \Delta t$ ;
            $(z_0, \dot{z}_0) = (z_1, \dot{z}_1)$ ;
        until $p_s$ periodic;
        save complete period of $z_0$ and $\hat{p}_\Gamma$ in $\mathbf{z}^n$ and $\hat{\mathbf{p}}_\Gamma^n$ ;
        calculate $\bar{F}_i$ from periodic solution $p_s$ ;
        adjust trim: $T_{i+1} = \mathcal{T}(T_i, \bar{F}_i)$ ;
    until $|\bar{F}_i - F_{target}| < \epsilon$;
    $t = 0$ ;
    $t = 0$ ;
    $z_0 = z_I,\ \dot{z}_0 = \dot{z}_I$ ;
    initialise flow state $U_0 = U_I$ ;
    repeat $j = 0, \dots$            // time stepping
        $U_j = \mathcal{F}\left(U_{j-1}, \mathbf{z}^n(j-1), \mathbf{z}^n(j)\right)$ ;
        calculate $p_\Gamma$ from $U_j^k$ ;
        $\mathbf{\Delta p}^{n+1}(j \bmod N_t) = p_\Gamma - \hat{\mathbf{p}}_\Gamma^n(j)$ ;
    until $p_\Gamma$ periodic;
    calculate $\bar{F}$ from periodic solution $U$ ;
until $|\bar{F} - F_{target}| < \epsilon$;
```

### Choice of simple model

The simplified model for the estimate of the pressure at the piston $\hat{p}_\Gamma$ must still be specified. A suitable approximation of the gas dynamics in the cylinder comes from the adiabatic expansion of an ideal gas. For the case that there is no heat exchange, e.g. if the piston is moved very rapidly, then the quantity $pV^\gamma$ is constant. With the initial pressure $p_I$ and initial fluid domain length $l_I$ as reference this gives the approximation

$$\hat{p}_\Gamma = p_I \left(\frac{l_I}{l}\right)^\gamma ,$$

with $l$ the average length of the cylinder in the current space–time slab.

When comparing the result of a tightly coupled procedure with the full CFD model and a coupling with this approximation, two problems become apparent. The first is that this simple model does not show a rise in temperature as great as seen in the CFD model. The second is that the tightly coupled procedure using the simplified model does not converge to a periodic solution due to a lack of damping, which is not a problem when using the CFD model. The damping in the CFD model is provided by the left boundary condition, where energy can flow in and out the fluid domain. As there is no damping in the structure itself, some form of damping from the fluid is necessary, because without damping the spring motion (which has a natural frequency different from the forcing motion) will not fade away and no periodic solution will be found.

To overcome these problems, the approximation is extended with two additional parameters. The purpose of these parameters is not to reflect certain physical properties of the CFD model, but rather to obtain a simplified model with a large-scale behaviour more similar to the CFD model.

$$\hat{p}_\Gamma = p_I \left( \frac{l_I - C_1}{l - C_1} \right)^\gamma - \dot{z} C_2 \tag{4.13}$$

$C_1$ induces greater pressure fluctuations because the cylinder of air seems smaller. This results in a higher average pressure at the piston, more close resembling the pressure found by the CFD procedure. The second parameter $C_2$ is to provide artificial damping. This parameter has no significant effect on the final periodic solution, but does decrease the number of periods in which it is attained. The best values for these parameters are experimentally found. In Section 4.5.2 the solution of this simplified model is compared to the CFD solution for a range of values for the trim parameter.

## 4.5. Trim adjustment

For the trim procedure two things are needed: a quantity derived from the solution of the combined fluid–structure system and an adjustable parameter that influences that quantity in a (more or less) predictable fashion.

In rotorcraft simulation for trimmed steady flight the quantities derived from the coupled solution are the forces and moments on the aircraft. For the purpose of trimming it is not necessary to know the exact loads over the whole length of the rotor blades. Likewise the time varying loads do not have to be known, but only the average over one rotor revolution. The trimmed quantity thus is generally chosen to be the mean rotor hub load. The equivalent quantity in the model problem is the mean aerodynamic force on the piston,

$$\bar{F} = \frac{1}{t_L} \int_0^{t_L} p_\Gamma A dt.$$

### 4.5.1. Choosing a trim parameter

Trim adjustments on rotorcraft are done by altering the prescribed angle of attack at the root of the rotor blades over the course of a whole rotor revolution. The periodic motion of the blades at the rotor hub is prescribed by some parameters that relate to the pilot controls. In general there are three degrees of freedom for the controls and three components of the rotor hub load.

As there is no adjustable parameter in the model problem that is obviously analogous to the angle of attack of the blade root, several possibilities are examined. The considered methods for trimming the fluid–structure model are the adjustment of the parameters describing the forced motion, $\bar{L}$ and $\Delta L$, and the parameters used in the structural model, $m$ and $k$. In Figure 4.3 the influence of adjusting each of these four parameters on $\bar{F}$ is shown.

The trimming process is easiest when the relation between the trim parameter and $\bar{F}$ is straightforward, preferably monotonic. For rotorcraft dynamics this is certainly not the case in the presence of aerodynamic effects like dynamic stall. However, for the analysis of the coupling methods, the relation for the model problem should be kept simple to avoid slow convergence of the trim procedure. A further requirement is that the simplified model shows the same general behaviour for a varying trim parameter as the CFD model. Of course the models are not expected to match exactly, but the simple model should at least be able to cover the same range of values for $\bar{F}$ as the CFD model in order for the loose coupling to be able to converge.

As can be seen in Figure 4.3(a), the force $\bar{F}$ is not a monotonic function of $\bar{L}$. The almost sawtooth-like shape is from pressure waves reflecting on the left fixed wall and arriving at the piston in or out of phase with the movement of the piston. As the value of $\bar{L}$ needed to attain a desired $\bar{F}$ is not uniquely defined, this is not a good choice for the trim parameter.

For lower range of values for $\Delta L$ a linear approximation for its influence on the force $\bar{F}$ is reasonably accurate. The sudden increase of $\bar{F}$ around $\Delta L = 0.12$ in Figure 4.3(b) is the result of the onset of local supersonic flow.

Figure 4.3(c) shows that the spring constant $k$ can be a good candidate for the trim adjustment parameter if its value is restricted to $k > 0.0025$. Preliminary tests have shown that this is indeed the case. Using $1/k$ as the trim parameter works even better, because then the response of $\bar{F}$ is more or less linear over a greater range.

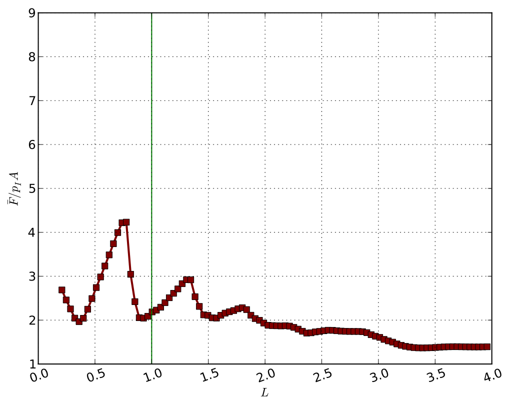
The last trimming strategy considered is varying the mass of the piston. This clearly has no meaningful equivalent in rotorcraft, but that is not important to consider for the model problem. The disadvantage of using $m$ as the trim parameter is that the range for which $\bar{F}$ depends more or less linearly on $m$ is very limited.

### 4.5.2. Effect of trim on the simple model

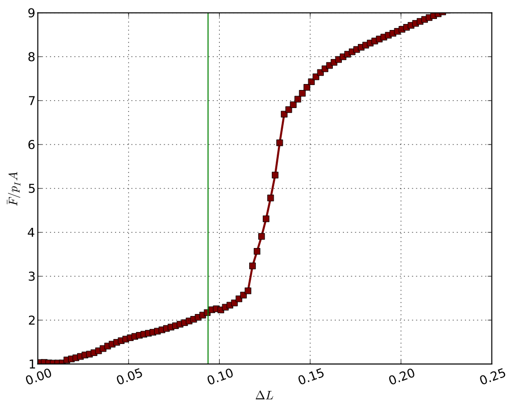
Based on the analysis in the previous section $\Delta L$ and $1/k$ are further investigated as possible trim parameters. To evaluate how well suited a parameter is for trimming in a loosely coupled procedure, the result of a tightly coupled procedure using the approximate model is compared to the result of a procedure using the CFD model for the same range of parameter values.

It can be seen in Figure 4.4 that although $1/k$ could be approximated linearly over a great range of values, the proposed simple model (4.13) does not result in a good approximation of the behaviour of $\bar{F}$ with respect to $1/k$ for any value of $C_1$. The response of the simple model to variations of $1/k$ is too flat, so a loosely coupled procedure with trimming on $1/k$ would not converge, as the difference between the two models is too big.

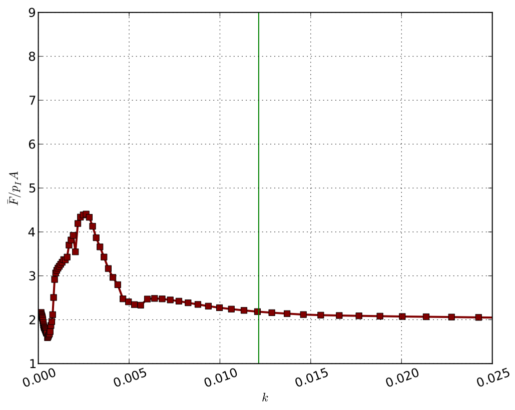
For variations of $\Delta L$ Figure 4.4(b) shows that the simplified model indeed fails to capture the subtleties of the behaviour of the CFD model. This is of course expected, but for $\Delta L < 0.12$ the simple model can be made to match the CFD results quite closely, which should result in good convergence behaviour for the loose coupling procedure. For larger values of $\Delta L$ the difference between the two models is so large that the loose coupling will most likely not converge at all,



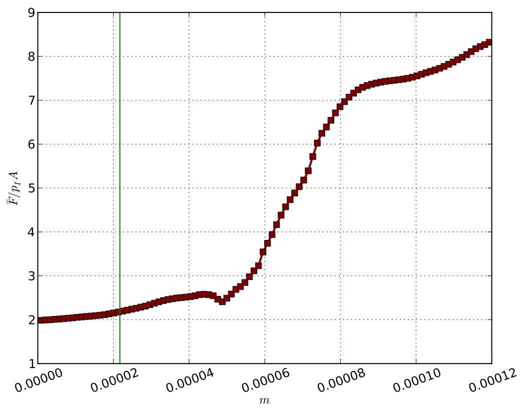
(a) $\bar{F}$ vs. $\bar{L}$ – using the mean value of the spring attachment point prescribed motion as trim parameter



(b) $\bar{F}$ vs. $\Delta L$ – using the amplitude If the spring attachment point prescribed motion as trim parameter



(c) $\bar{F}$ vs. $k$ – using the spring constant as trim parameter

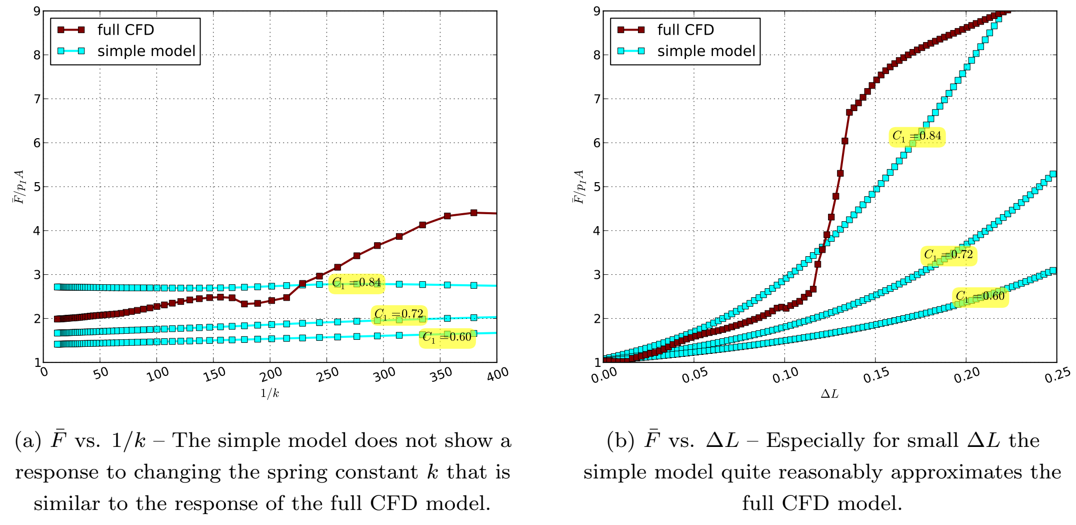


(d) $\bar{F}$ vs. $m$ – using the piston mass as trim parameter

**Figure 4.3.:** Influence of several possible choices for the trim parameter on the mean aerodynamic force $\bar{F}$ on the piston. The green line signifies the value for the parameter that is used when one of the others is varied, this is the value from Table A.1 in nondimensional form.

but that is not necessarily a problem.

Of the four considered possibilities for choice of trim parameter, $\Delta L$ is the most likely to lead to a successful loose coupling. Therefore the trim procedure $\mathcal{T}$ will adjust $\Delta L$ in order to search for an average force on the piston $\bar{F}$ that is equal to $F_{\text{target}}$.



(a) $\bar{F}$ vs. $1/k$ – The simple model does not show a response to changing the spring constant $k$ that is similar to the response of the full CFD model.

(b) $\bar{F}$ vs. $\Delta L$ – Especially for small $\Delta L$ the simple model quite reasonably approximates the full CFD model.

**Figure 4.4.:** Comparison of three versions of the simplified model to the solution of the full CFD model for two different choices of the trim parameter

### 4.5.3. Trim solution strategy

In steady forward flight the aircraft is said to be trimmed when the controls are such that the mean total aerodynamic forces are equal to the weight and drag of the aircraft. The rotor hub moments should be zero, with the exception of the yawing moment, which can be compensated by the tail rotor in most rotorcraft configurations. Every degree of freedom of the controls has a nontrivial influence on the rotor hub loads. To keep the trimming process simple, a linear approximation with a fixed gradient is used. To estimate the Jacobian of the relation between input (trim) and output (loads) a $3 \times 3$ matrix is constructed called the sensitivity matrix. Each column of this matrix consists of the difference of the loads resulting from a small perturbation of one of the trim inputs with a reference load. The inverse of the sensitivity matrix is used to determine in which direction the trim parameters should be adjusted in order to get closer to the desired rotor hub loads.

For the model problem the trimming process consists of finding the right value for $\Delta L$ for which the solution of the combined fluid–structure system yields an $\bar{F}$ that is equal to some chosen target. In this simple case the sensitivity matrix is a scalar and is denoted by $\delta\bar{F}$. It is determined by two runs of the coupled model, for different, fixed, values of the trim parameter $T_0$ and $T_1$. For subsequent trim iterations, the new value for the trim parameter, $T_{i+1}$, is determined by linear extrapolation of the last parameter value $T_i$ and its resulting force $\bar{F}_i$ using the sensitivity matrix.

$$\begin{aligned} T_{i+1} &= T_i + \frac{1}{\delta \bar{F}} \left( F_{\text{target}} - \bar{F}_i \right) \qquad \text{for } i = 1, 2, \ldots, \\ \delta \bar{F} &= \frac{\bar{F}_1 - \bar{F}_0}{T_1 - T_0}. \end{aligned}$$

Although for the model problem the (1-dimensional) sensitivity matrix could be updated each trim iteration, it is kept fixed in order to keep the procedure similar to the rotorcraft trimming procedure.

CHAPTER 5

# Evaluation of the model problem and coupling strategies

## 5.1. Untrimmed FSI coupling

### 5.1.1. Convergence to a periodic solution

In Section 4.4.1 it is assumed that after starting from some initial condition the transients will disappear and a periodic solution will be obtained after some time. To validate this, a flow solution is calculated for multiple periods by the combined fluid–structure model with the trim kept fixed.

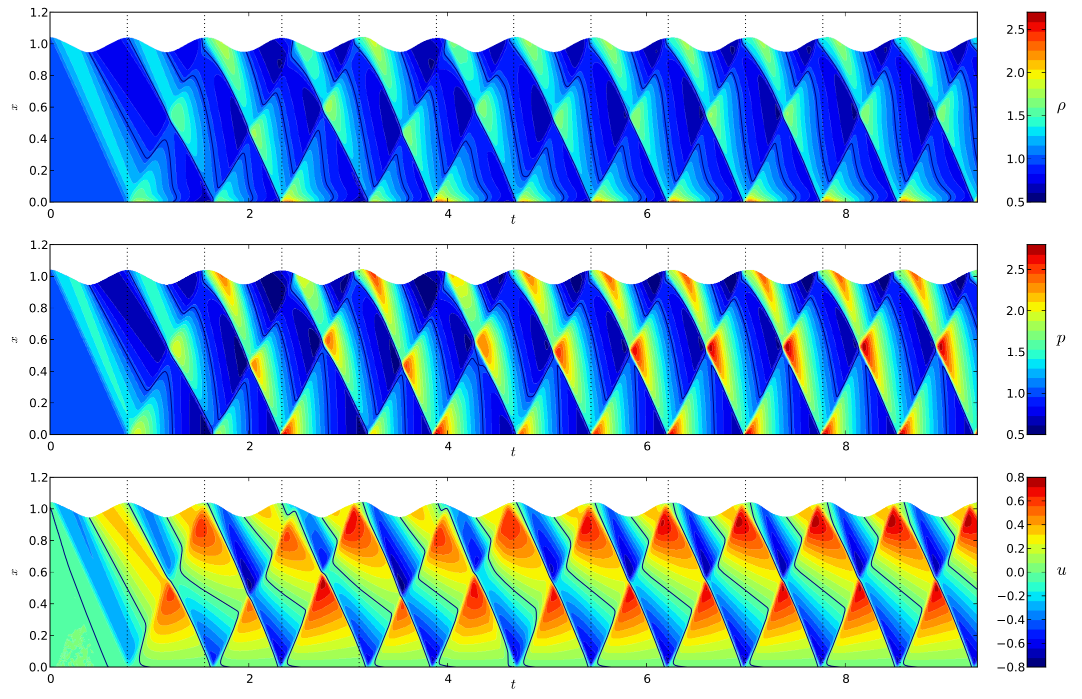
In Figure 5.1 the flow solution for 12 periods of the forcing motion is shown for two different values of the trim parameter $\Delta L$. The flow variables $\rho$, $u$ and $p$ are shown separately for the whole space–time domain. Time is on the horizontal axis and the length along the cylinder on the vertical axis. The oscillating motion of the piston is reflected by the curved upper boundary of the domain.

Starting from the initial fluid state, perturbations of the mean flow state emerge at the moving boundary and travel to the left side of the piston, where they are reflected. After several periods the fluid does indeed seem to attain a periodic solution. The main difference between the flow states for the two values of $\Delta L$ is that the larger $\Delta L$ results in a greater movement of the piston and larger perturbations for the flow variables.

The simplified model does not calculate a complete fluid state, but only consists of an expression for the pressure at the piston $p_\Gamma$. A common expression for the periodicity of the solution is desired, so $p_\Gamma$ will be used to determine the periodicity of the solution for both the CFD model and the simple model. Periodicity of $p_\Gamma$ does not necessarily mean that the whole fluid state is periodic, but as this pressure at the interaction boundary is the only information from the CFD model needed for the fluid-structure coupling, it suffices to measure periodicity this way.

Figure 5.2 shows the pressure on the piston for the same CSD-CFD model parameters as before, over 20 periods of the forced motion. After a strongly non-periodic start the pressure profile settles around $t = 6$, after 8 periods. During the next periods the pressure profile does not change much anymore.

As was already shown in Section 4.5.1, the average pressure increases when $\Delta L$ increases. The value of the average pressure at the piston (the red line in Figure 5.2) converges to the value found in Figure 4.3(b) for the two trim parameter values. In the periodic pressure profile for $\Delta L = 0.11$ a secondary peak can be seen that is not present in Figure 5.2(a). This is due to the reflected wave hitting the piston on the start of the outgoing movement, whereas in the case of


(a) $\Delta L = 0.04$

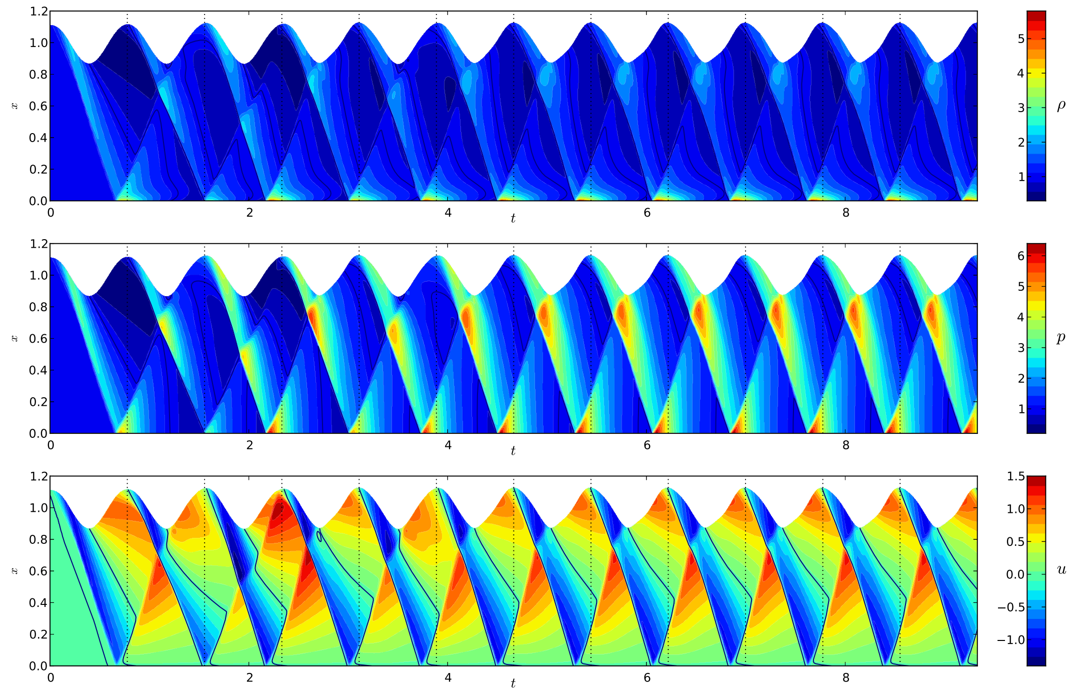
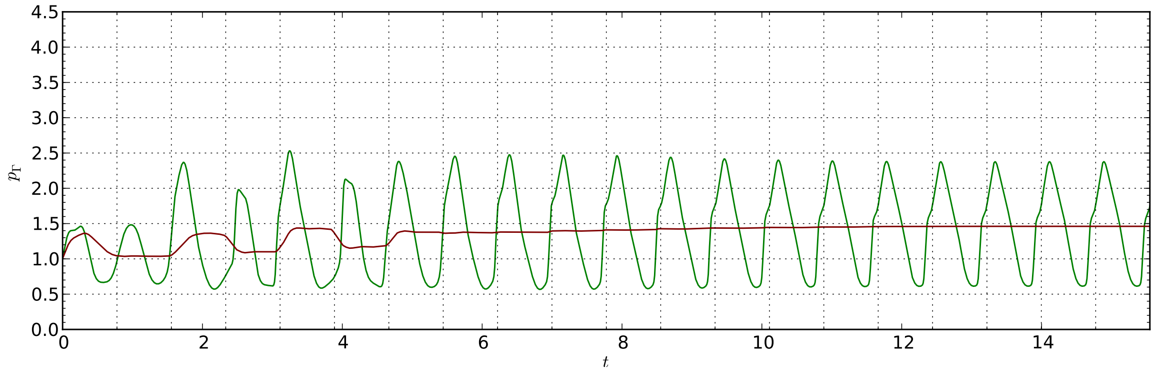
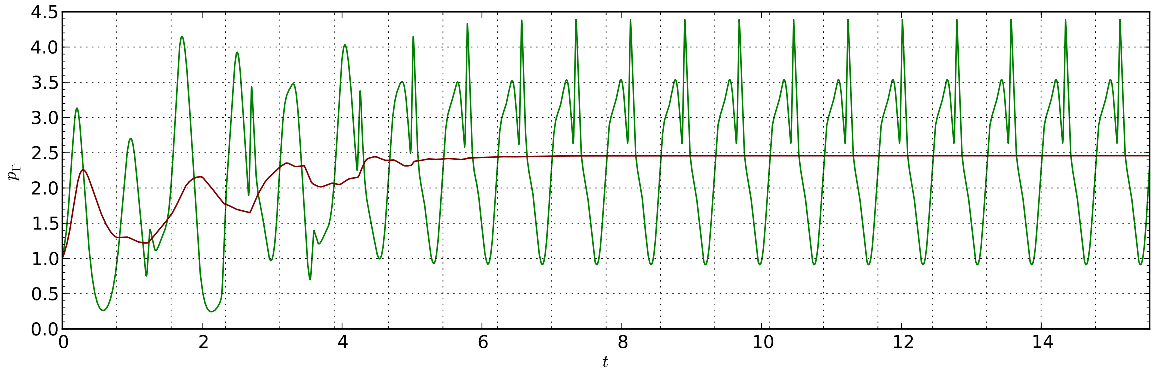

(b) $\Delta L = 0.11$

**Figure 5.1.:** Fluid flow for 12 periods of the CFD model for two different values of the trim parameter $\Delta L$. The density, fluid velocity and pressure are shown in the space–time domain with the length along the cylinder on the vertical axis. Time is on the horizontal axis.



(a) $\Delta L = 0.04$



(b) $\Delta L = 0.11$

**Figure 5.2.:** Pressure at the piston for 20 periods of the CFD model.
The moving average for one period is shown in red. For $\Delta L = 0.11$ the average pressure converges to a higher value and the periodic pressure profile has a secondary peak.

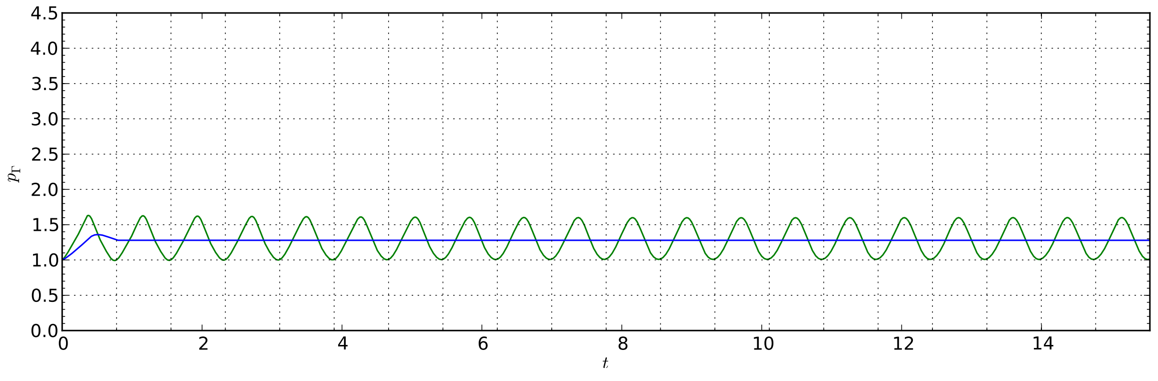
$\Delta L = 0.04$ the reflection coincides with pressure peak resulting from the ingoing motion of the piston.

### 5.1.2. Comparison with the simple model

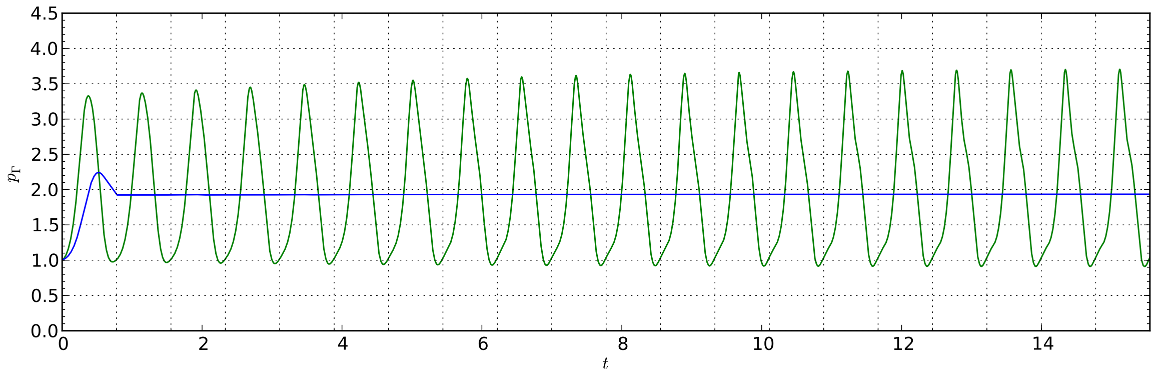
Figure 5.3 shows the pressure on the piston as calculated by the simplified model. For the approximation the simple model is coupled to the CSD model using the same tight coupling with fixed trim as with the full CFD model. This procedure is essentially the same as the inner loop of the loose coupling. The only difference is that there is no $\Delta p$ correction applied.

As already shown in Section 4.5.2, the simple model with $C_1 = 0.72$ underestimates the value of $p_\Gamma$, but does capture the relation with the trim parameter $\Delta L$. The shape of the pressure profile is much simpler compared to the result of using the full CFD model. The general behaviour, however, matches that of the pressure calculated by the CFD model.

For illustration purposes the damping parameter $C_2$ used in Figure 5.3 has been reduced by a factor of ten to make the convergence to a periodic solution (especially for the case $\Delta L = 0.11$) clearly visible. When the normal value for $C_2$ is used, the pressure profile looks periodic after only 2 periods, as is the case in Figure 5.3 for $\Delta L = 0.04$.



(a) $\Delta L = 0.04$



(b) $\Delta L = 0.11$

**Figure 5.3.:** Pressure at the piston for 20 periods of the simplified model. ($C_1 = 0.72$) The moving average for one period is shown in red. Here also a greater prescribed motion leads to a larger average pressure.

To better quantify the convergence to a periodic resolution, Figure 5.4 shows the difference between $p_\Gamma$ at time $t$ and the pressure exactly one period before, plotted on a logarithmic scale. If the models truly converge to a periodic solution, this periodic difference should go to zero. The most important observation from this plot is that the CFD solution does not actually converge to a periodic solution. Where the simple model shows steady convergence to a periodic solution, the CFD model has reached maximum periodicity after about 12 periods. In practice the attained periodicity of $10^{-4}$ may or may not be enough to give meaningful results. For the current research it suffices though, because the emphasis is on the convergence of the outer iterations of the loose coupling, not on the convergence to a periodic solution.

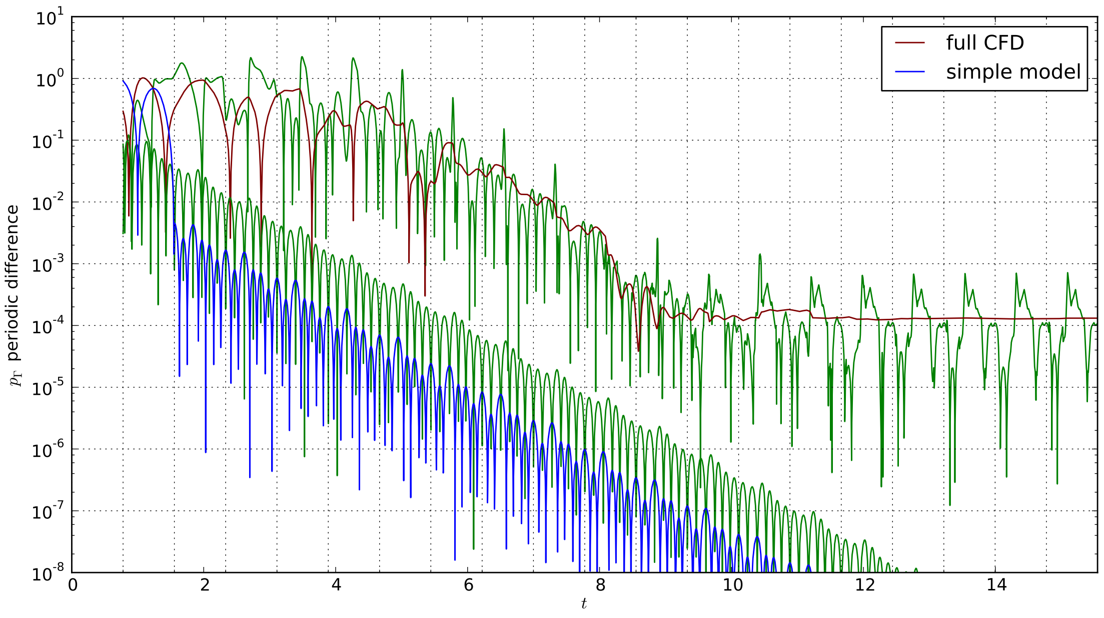


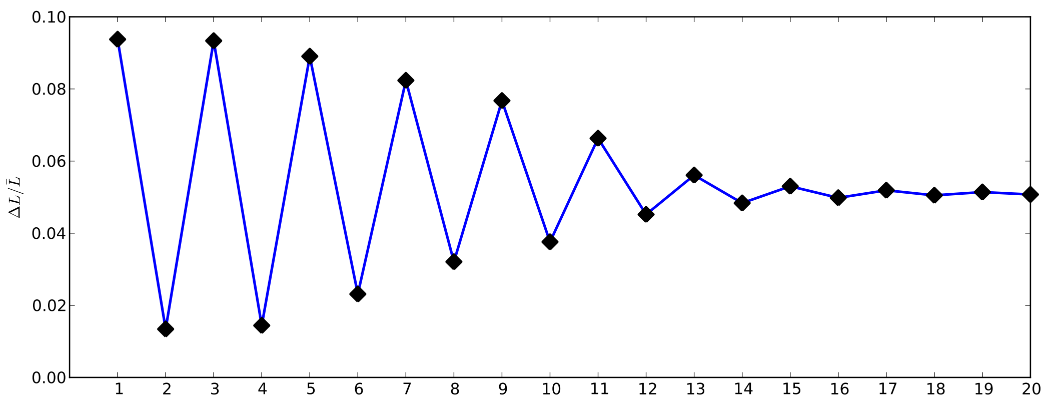
**Figure 5.4.:** Periodic difference of 20 periods of the pressure at the piston for the case $\Delta L = 0.11$. The periodic difference of $p_\Gamma$ is plotted in green and the periodic difference of the one period average is plotted in red and blue for the CFD and simple model respectively.

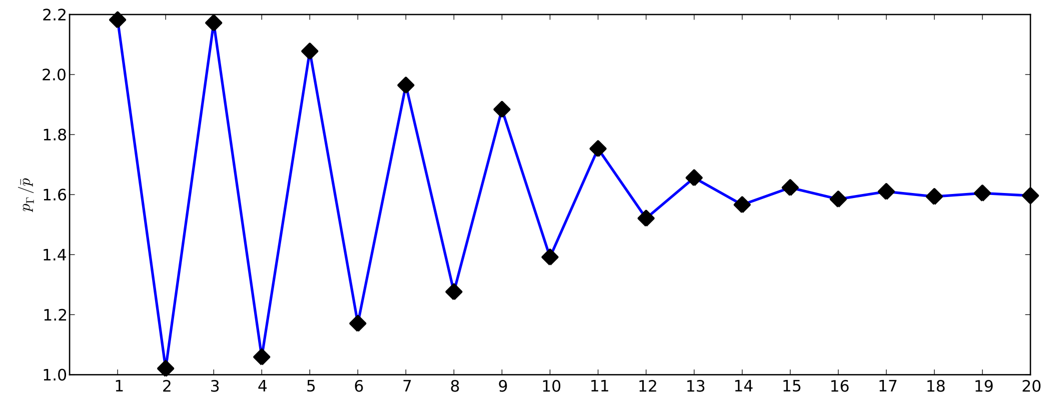
## 5.2. Coupled model with trimming

### 5.2.1. Tight coupling

Finding the right value of the trim parameter for a specified target $p_\Gamma$ involves multiple iterations of the coupled CSD-CFD model with different trim parameter values. Figure 5.5 shows the tight coupling converging to the $\Delta L$ required to reach the target. Each iteration shows the average pressure at the piston of the periodic solution for a single value of the trim parameter. The first two iterations, which are necessary to determine the sensitivity matrix, are not shown.

The convergence of $p_\Gamma$ shows oscillatory behaviour. In Figure 5.5(c) it can be seen that the convergence is very slow for the first 10 iterations and somewhat better after that. The exact convergence behaviour for the tight coupling is highly dependent on the chosen initial values for $\Delta L$ to determine the sensitivity matrix. In this case the sensitivity matrix is determined around 0.02, where the $p_\Gamma/\Delta L$ gradient is quite flat. If the sensitivity matrix would be determined


(a) Evolution of the trim parameter $\Delta L$ throughout the iterations


(b) One period average of $p_\Gamma$ converging to 1.6

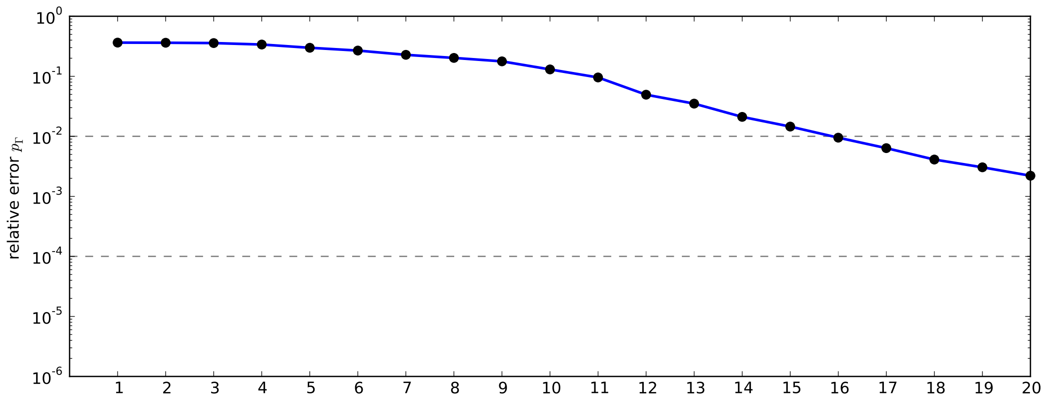

(c) Relative error of $p_\Gamma$ on a logarithmic scale

**Figure 5.5.:** 20 iterations of the tightly coupled procedure (Algorithm 1) for finding the trim for $p_\Gamma = 1.6$ times the initial pressure in the cylinder

around 0.05, convergence would be much better, but obviously the right value is not known in advance, as it is the result of the trimming process.

### 5.2.2. Loose coupling

Figure 5.6 shows the convergence history of the loosely coupled procedure. The outer loop of the loose coupling consists of finding a trimmed solution using the simple model, followed by a single run of the CFD model with prescribed piston motion. The 20 CFD steps are plotted using a solid marker. Each CFD step is preceded by several iterations of the simple model, plotted with open markers.

For the loose coupling specified in Algorithm 2, the number of simple model runs is based on how close the resulting pressure is to the target value. For the easy reading of the plots, here a fixed number of 10 simple model runs is used. This is more than is necessary to get a good estimate of $p_\Gamma$, but as the simple model is not very computationally expensive, using a fixed number is not a problem.

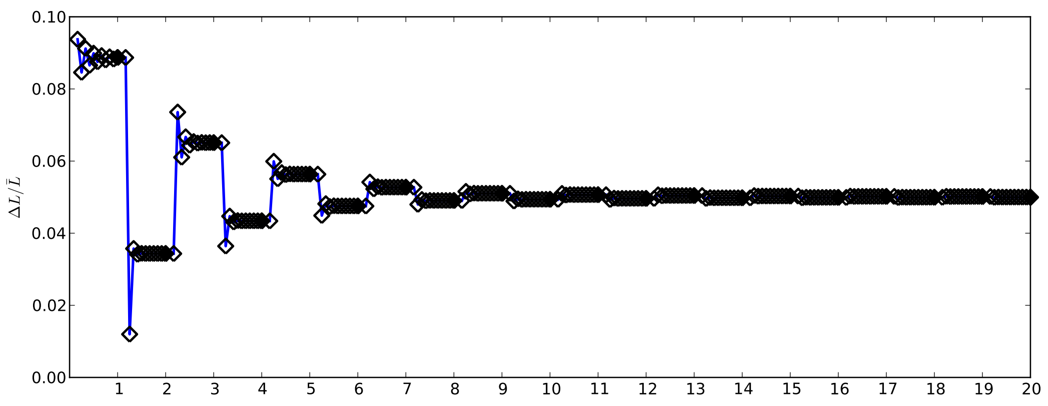
There are two types of oscillations visible in Figure 5.6. On a small scale there are the oscillations resulting from trimming the simple model. As was the case with the tight coupling, these oscillations appear because the sensitivity matrix is determined in a region where the simple model has a flatter $p_\Gamma/\Delta L$ gradient compared to the region of the desired trim parameter value. On a larger scale the oscillations occur after each CFD step. This is the result of the simple model underpredicting the influence of the trim parameter on $p_\Gamma$. In this case the simple model first finds that $\Delta L = 0.09$ is needed to attain a $p_\Gamma$ of 1.6. The CFD correction step shows that this value of the trim parameter results in a pressure that is too large, around 2.0. Subsequently $\Delta p$ is adjusted and the simple model is trimmed again, this time converging to $\approx 0.035$. Now the CFD correction results in a lower pressure. This results in the stepwise oscillatory behaviour seen in Figure 5.6(a).

Comparing the tight and the loose coupling after 20 CFD iterations shows that they attain about the same relative error. It is important to note though, that the computational costs of finding a periodic CFD solution for the loose coupling are about 2–3 times less. This is because in the case of the loosely coupled model the length of the fluid domain is prescribed for the CFD model run, so there is no interaction with the CSD model during the CFD computations and consequently no subiterations are necessary to achieve a coupled fluid–structure solution. The time needed to calculate a periodic solution using the simple model is negligible, so in terms of computational cost the loose coupling can be said to converge faster than the tightly coupled procedure.

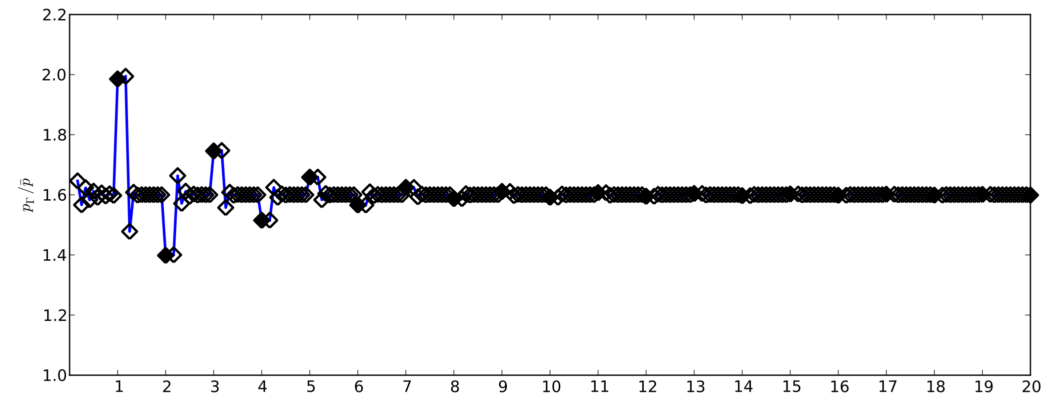
An important observation is that the trim parameter found is the same ($\Delta L \approx 0.0484$), so the loose coupling is consistent with the tightly coupled procedure. It must be stressed that, although a relative error of $10^{-3}$ would probably suffice for a rotorcraft aerodynamics simulation for engineering purposes, the goal for this model problem, which is much simpler, is to obtain better accuracy and faster convergence.

#### Convergence of periodic $\mathbf{p}_\Gamma$

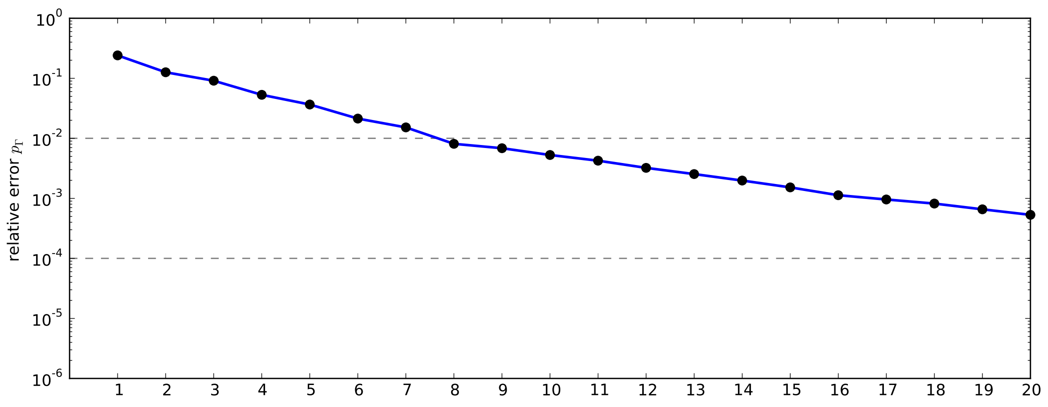
It is informative to look at how the periodic pressure at the piston evolves during the outer iterations of the loose coupling. Figure 5.7(a) shows the pressure at the piston as found by the



(a) Evolution of the trim parameter $\Delta L$ throughout the iterations

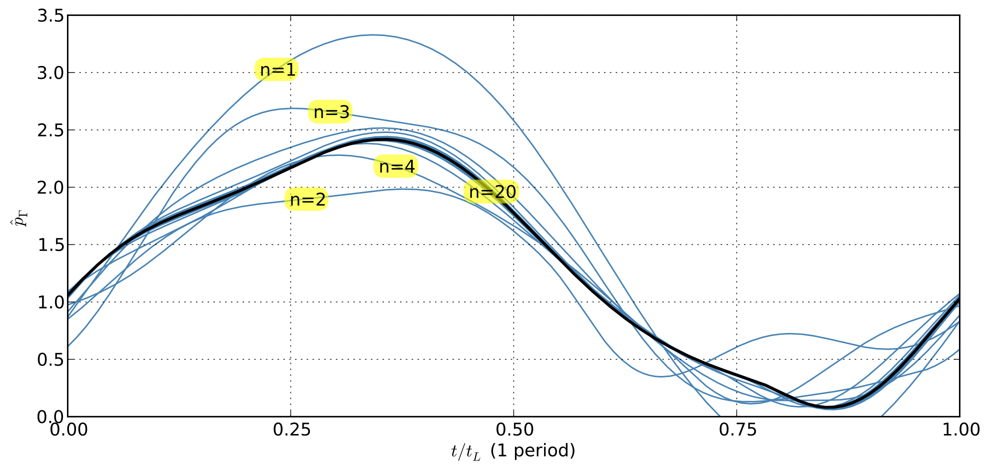


(b) One period average $p_\Gamma$. Solid diamonds are the CFD steps.

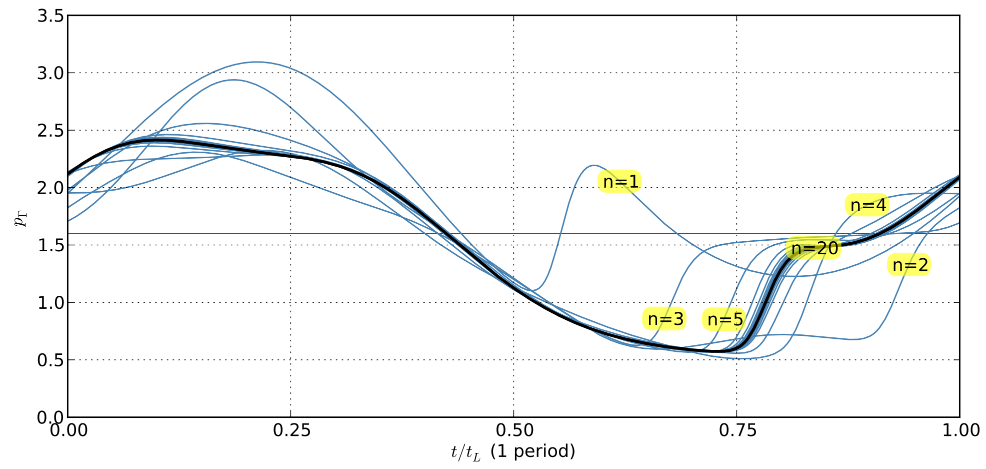


(c) Relative error of $p_\Gamma$ on a logarithmic scale, after each CFD step

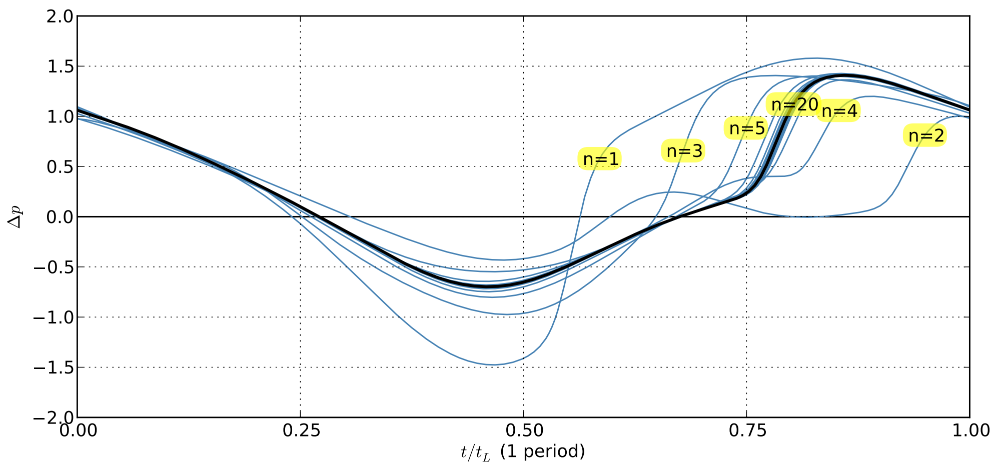
**Figure 5.6.:** 20 iterations of the loosely coupled procedure (Algorithm 2) for finding the trim for $p_\Gamma = 1.6$ times the initial pressure in the cylinder



(a) Pressure calculated by the simplified model



(b) Pressure calculated by the CFD model



(c) Difference between $\mathbf{p}_\Gamma$ and $\hat{\mathbf{p}}_\Gamma$

**Figure 5.7.:** Fluid pressure at the piston over a whole period. The first five and the last of the iterations are labelled.

simple model. It converges to a fairly straightforward pressure profile. The first iteration has a section with negative pressure. Here the simple model fails to give a physically meaningful result. This does not lead to stability problems for the fluid–structure interaction however, because the fluid pressure is only used to calculate the force on the piston. A negative fluid pressure means the fluid exerts a force on the piston in the same direction as the force from the atmospheric pressure $p_a$ outside the cylinder. It also does not affect the final outcome or the stability of the trim procedure, because the coupling uses the average pressure as input to the trim procedure, which remains positive.

Figure 5.7(b) shows $\mathbf{p}_{\boldsymbol{\Gamma}}$ of the periodic solution calculated by the CFD model. The solution converges to a pressure profile with the average of the target pressure $p_\Gamma = 1.6$. Finally, Figure 5.7(c) shows $\boldsymbol{\Delta}\mathbf{p}$ for 20 iterations of the loose coupling. For the middle part of the period it is negative, meaning that in this region the simple model over predicts the fluid pressure compared to the CFD model. The average of $\boldsymbol{\Delta}\mathbf{p}$ converges to a positive value, consistent with the earlier observations that the simple model under predicts $p_\Gamma$.

CHAPTER 6

# Improving the loose coupling convergence

In this chapter some strategies for improving the convergence of the loose coupling will be explained and evaluated. The focus is on reducing the number of CFD solutions necessary to get a fully coupled trimmed periodic solution.

## 6.1. Relaxation

A well known approach to stabilise a coupled iterative procedure and speed up its convergence is to apply some amount of relaxation to the updates of the variables involved. In the loosely coupled procedure there are two variables that are updated in the outer loop that are of importance. The result of the trimming process using the simple model is the periodic piston motion $\mathbf{z}$, which is used to prescribe the length of the fluid domain in the subsequent CFD iterations. The periodic solution of the CFD model yields the pressure difference $\mathbf{\Delta p}$. This vector is then used as a correction to the pressure calculated by the next round of the simple model. Both relaxation on $\mathbf{z}$ and on $\mathbf{\Delta p}$ are examined.

### 6.1.1. Relaxation on $\mathbf{\Delta p}$

The pressure difference $\mathbf{\Delta p}$ can be written as a recurrence relation by combining Equations (4.12) and (4.11)

$$\begin{aligned} \mathbf{\Delta p}^{n+1} &= \mathbf{p}_{\mathbf{\Gamma}}^{n} - \hat{\mathbf{p}}_{\mathbf{\Gamma}}^{n} \\ &= \mathbf{p}_{\mathbf{\Gamma}}^{n} - \mathbf{p}_{\mathbf{s}}^{n} + \mathbf{\Delta p}^{n}. \end{aligned} \tag{6.1}$$

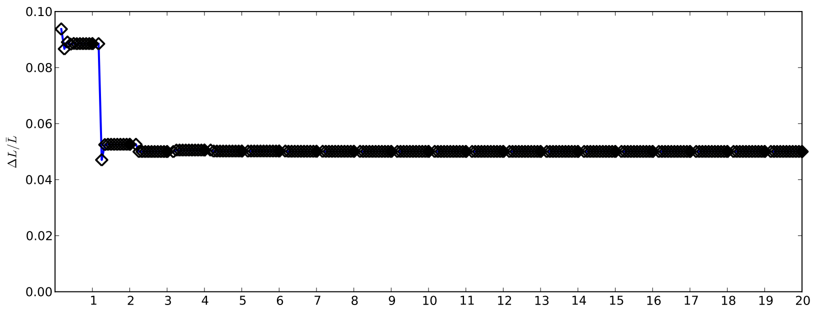
As can be seen in Figure 5.6 the simple model initially underestimates the pressure at the piston. After the CFD model has calculated the periodic solution, $\mathbf{\Delta p}$ is updated to reflect the difference. When the simple model is used again to trim $p_s$ back to the target value, the next run of the CFD model shows that the pressure was overestimated. $\mathbf{\Delta p}$ now needs to be adjusted downwards.

The oscillatory behaviour of $p_\Gamma - p_s$, the average of the updates to $\mathbf{\Delta p}$, suggests that relaxed updating of $\mathbf{\Delta p}^n$ could be beneficial to the convergence of the loose coupling. When relaxation is applied with a parameter $0 < \phi \leq 1$, Equation (6.1) becomes

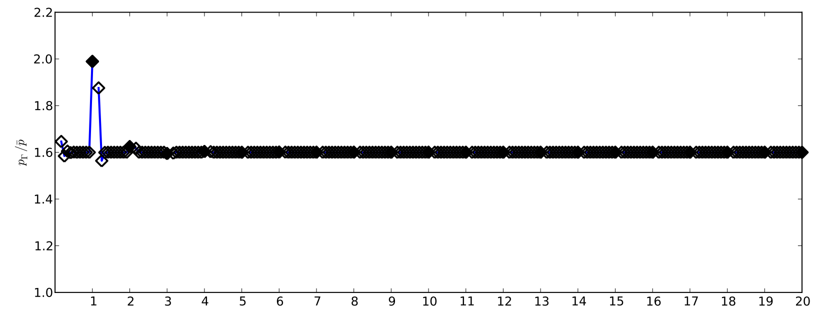
$$\begin{aligned} \mathbf{\Delta p}^{n+1} &= \phi \left(\mathbf{p}_{\mathbf{\Gamma}}^{n} - \hat{\mathbf{p}}_{\mathbf{\Gamma}}^{n}\right) + (1-\phi)\mathbf{\Delta p}^{n} \\ &= \phi \left(\mathbf{p}_{\mathbf{\Gamma}}^{n} - \mathbf{p}_{\mathbf{s}}^{n}\right) + \mathbf{\Delta p}^{n}, \end{aligned} \tag{6.2}$$

which for $\phi = 1$ results in the old update rule.

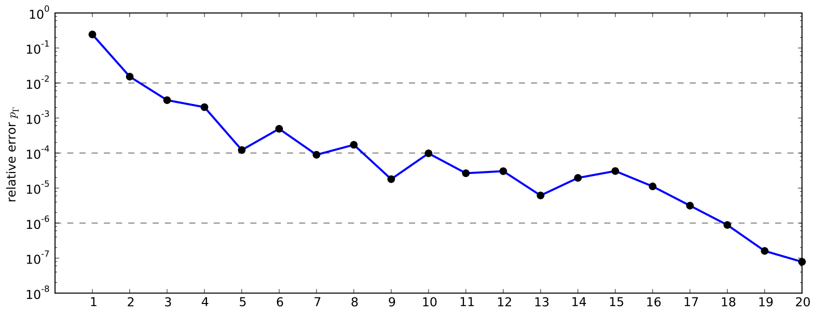
Figure 6.1 indeed shows that convergence is much faster with relaxation applied to the updates of $\mathbf{\Delta p}$. Where the unrelaxed loose coupling required 17 runs of the CFD model to achieve a relative error of $10^{-3}$, the method using relaxation only needs 5 CFD solutions for the same



(a) Evolution of the trim parameter $\Delta L$ throughout the iterations



(b) One period average $p_\Gamma$. Solid diamonds are the CFD steps.



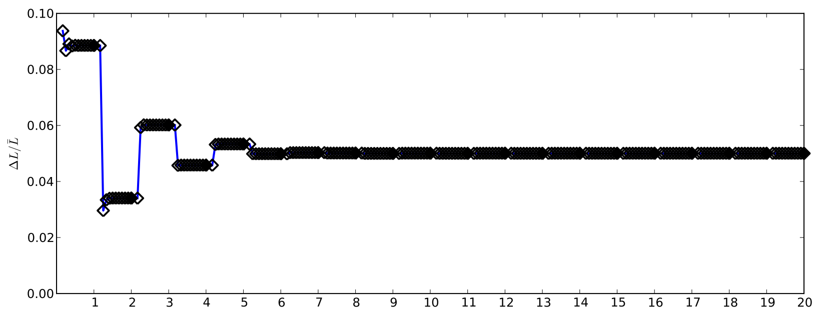
(c) Relative error of $p_\Gamma$ after each CFD step, on a logarithmic scale

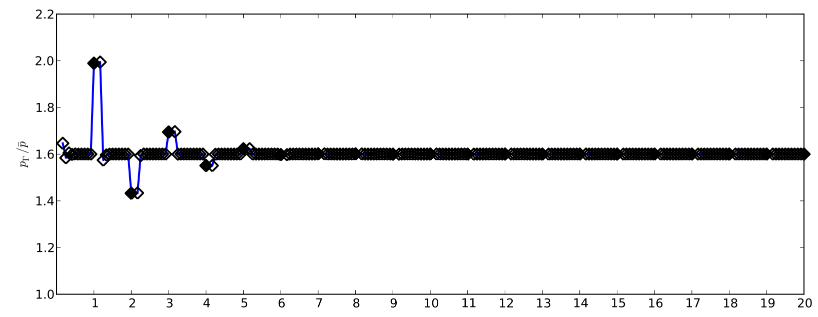
**Figure 6.1.:** 20 iterations of the loosely coupled procedure with relaxation on $\mathbf{\Delta p}$ for $\phi = 0.7$

error. The relaxation parameter $\phi$ is set here to a fixed value of 0.7, which is experimentally determined to yield good results for this instance of the model problem.

When looking at the results for more than 20 iterations of the loose coupling (Figure 6.6), it can be seen that without relaxation the loose coupling does not converge beyond a relative error of about $10^{-3}$. This is because the chosen parameter values for the simple model result in a model that does not show a behaviour similar enough to the full CFD model. Relaxation can help in this case however. For all the tested relaxation parameters, there was no lower bound on the relative error reached, other than the machine precision of the floating point number representation.

### 6.1.2. Relaxation on $\mathbf{z}$


(a) Evolution of the trim parameter $\Delta L$ throughout the iterations


(b) One period average $p_\Gamma$. Solid diamonds are the CFD steps.

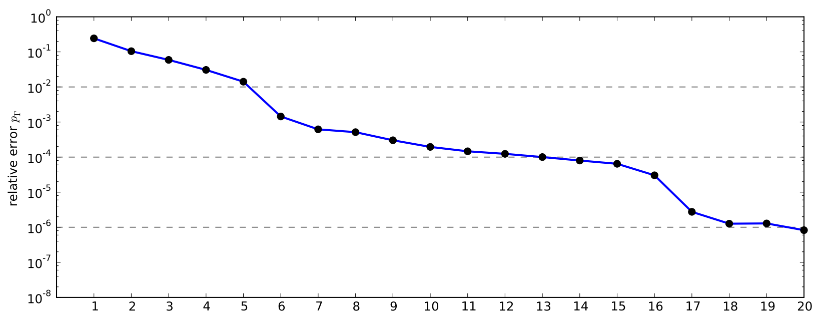
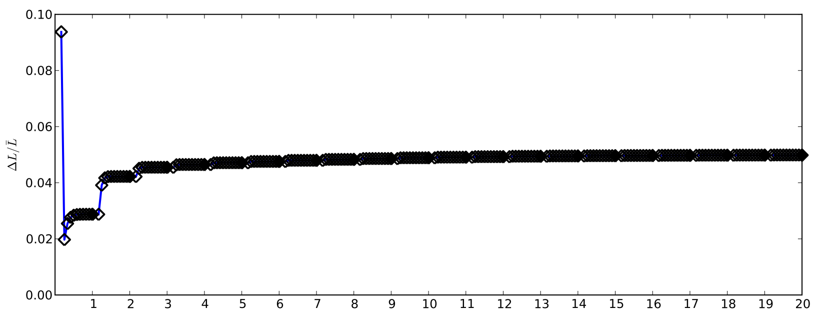

(c) Relative error of $p_\Gamma$ after each CFD step, on a logarithmic scale

**Figure 6.2.:** 20 iterations of the loosely coupled procedure with relaxation on $\mathbf{z}$ for $\phi = 0.95$

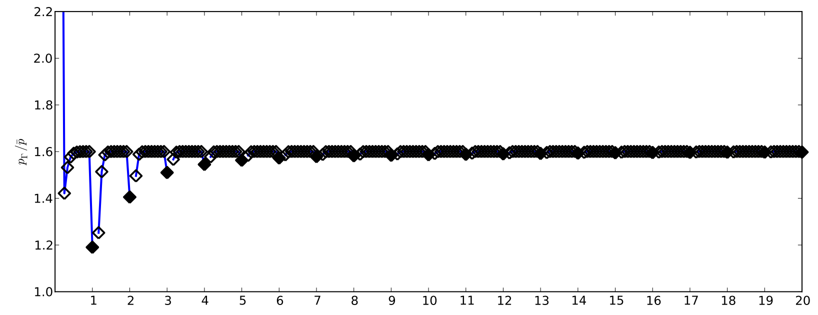
Instead of relaxing the updates of $\mathbf{\Delta p}$, another possibility is to apply the same method to $\mathbf{z}$. The resulting procedure also shows an improved convergence rate, as can be seen in Figure 6.2 for $\phi = 0.95$.

The important difference between the two methods of relaxation is the range of parameter values over which they produce improved results. For relaxation on $\mathbf{\Delta p}$ any strategy with a fixed $\phi$ where $0.5 < \phi < 1$ yields a better procedure than the loose coupling baseline, as can be seen in Figure 6.6. In the case of relaxation on $\mathbf{z}$, only for $0.9 < \phi < 1$ convergence improves. For $\phi < 0.9$ convergence is actually slower than the unrelaxed case. Therefore relaxation on $\mathbf{\Delta p}$ seems a better approach and from now on the focus will be on relaxation on $\mathbf{\Delta p}$.

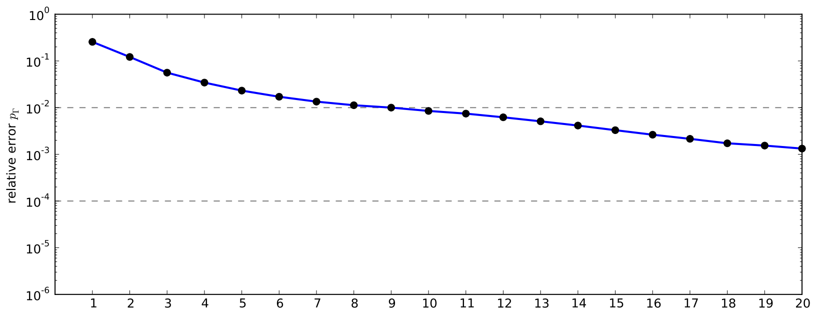
### 6.1.3. Disadvantages of relaxation with a fixed parameter



(a) Evolution of the trim parameter $\Delta L$ throughout the iterations



(b) One period average $p_\Gamma$. Solid diamonds are the CFD steps.



(c) Relative error of $p_\Gamma$ after each CFD step, on a logarithmic scale

**Figure 6.3.:** 13 iterations of the loosely coupled procedure with the simplified model using $C_1 = 0.89$ instead of 0.69

In the previous examples, the difference between the pressure found by the simplified model and the CFD model oscillates between a positive and a negative adjustment. This is because the chosen simplified model has a lower $\partial\bar{F}/\partial\Delta L$ gradient than the CFD model. As it is in general unknown which configuration of the simplified model leads to the best prediction of the CFD behaviour, it is instructive to look at the loose coupling using a simplified model with a different value for the parameter $C_1$. The simplified model is changed to react more strongly on changes in $\Delta L$, by using $C_1 = 0.89$. Figure 6.3 shows the results of the unrelaxed loose coupling with this adjusted simplified model. In this case the $\Delta p$ corrections after each CFD step are all negative and instead of oscillating, the found trim parameter $\Delta L$ increases monotonically.

When there are no oscillations in the CFD correction steps, relaxation using a fixed parameter with a value of $\phi < 1$ does not help to speed up convergence, but rather slows it down, as can be seen in Figure 6.4. A value of $\phi > 1$ could be used, but as this can easily lead to an unstable procedure, this approach is not further investigated.

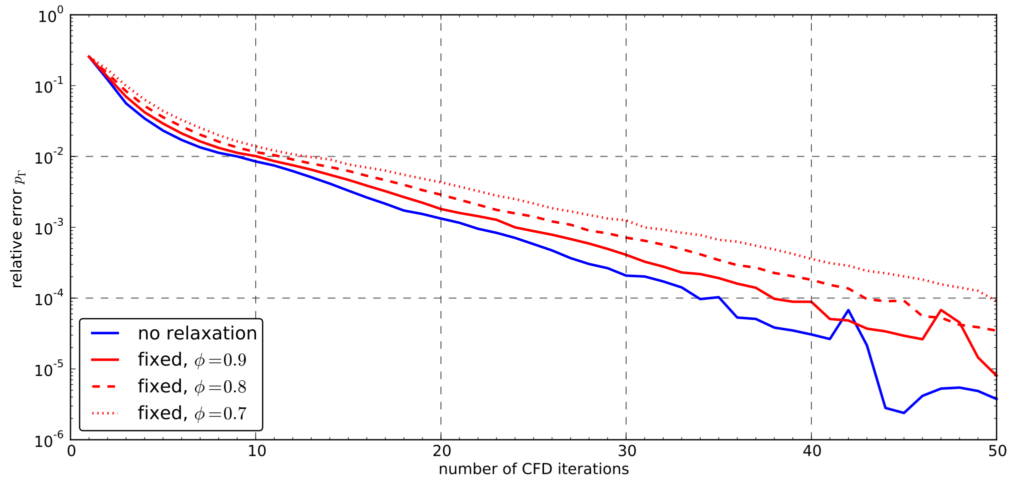


**Figure 6.4.:** 50 iterations of the loosely coupled procedure with the simplified model using $C_1 = 0.89$ instead of 0.69, using relaxation on $\mathbf{\Delta p}$ with fixed $\phi$

In general, success of the relaxation strategy is highly dependent on the choice of $\phi$, as can be seen in Figures 6.4 and 6.6 . The best choice of $\phi$ depends on the model parameters and how closely the simple model matches the CFD model. This makes it difficult to have a successful relaxation strategy using any fixed $\phi$ that is robust for a large range of model parameters.

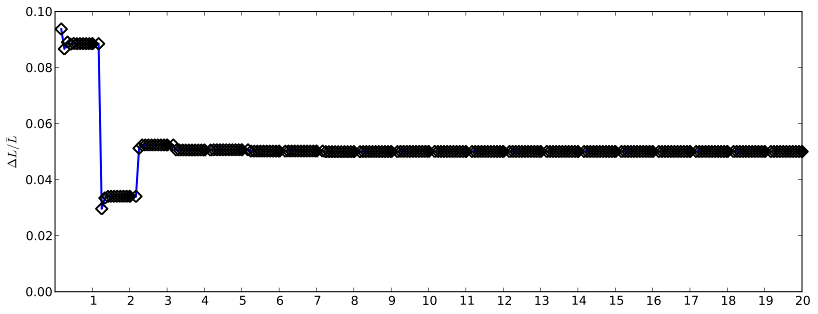
### 6.1.4. Aitken relaxation

In order to remove the need for setting $\phi$ directly, a dynamical approach to setting the relaxation parameter is used. A suitable strategy for determining $\phi$ is Aitken relaxation, as described by Küttler and Wall[17]. The method uses the previous two iterations to determine the new value of the relaxation parameter.

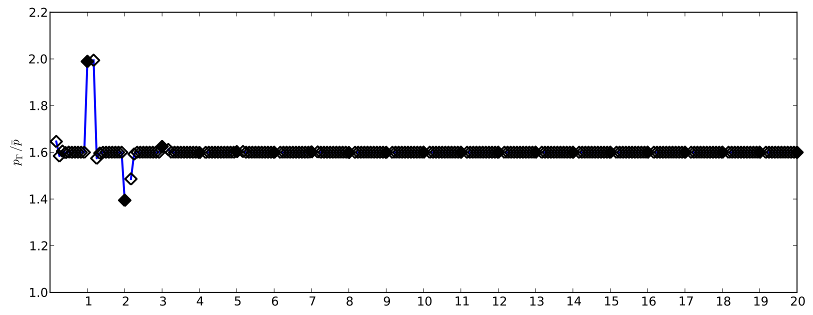
$$\phi^n = \phi^{n-1} \frac{(\mathbf{P}^n)^T \left(\mathbf{P}^{n+1} - \mathbf{P}^n\right)}{\left(\mathbf{P}^{n+1} - \mathbf{P}^n\right)^T \left(\mathbf{P}^{n+1} - \mathbf{P}^n\right)},$$

$$\text{with} \qquad \mathbf{P}^{n+1} = \mathbf{p}^n_\Gamma - \mathbf{p}^n_\mathbf{s}$$

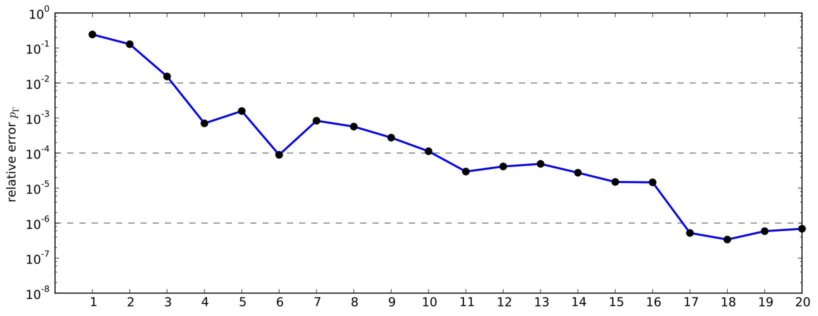
For the update after the first iteration $\phi^0$ is set to 1.



(a) Evolution of the trim parameter $\Delta L$ throughout the iterations



(b) One period average $p_\Gamma$. Solid diamonds are the CFD steps.



(c) Relative error of $p_\Gamma$ after each CFD step, on a logarithmic scale

**Figure 6.5.:** 20 iterations of the loosely coupled procedure with Aitken relaxation on $\mathbf{\Delta p}$

The results for the application of Aitken relaxation to the loose coupling are shown in Figure 6.5. It performs better than relaxation with fixed $\phi$.

During the coupling iterations $\phi^n$ is mostly set to values in the range $0.4 - 1.0$, but with some outliers as high as 1.4. Because the dynamically determined value for the relaxation parameter can also be greater that 1 if necessary, Aitken relaxation also improves the convergence speed when the simplified model with $C_1 = 0.89$, as described in Section 6.1.3, is used.

### 6.1.5. Summary of results for relaxation

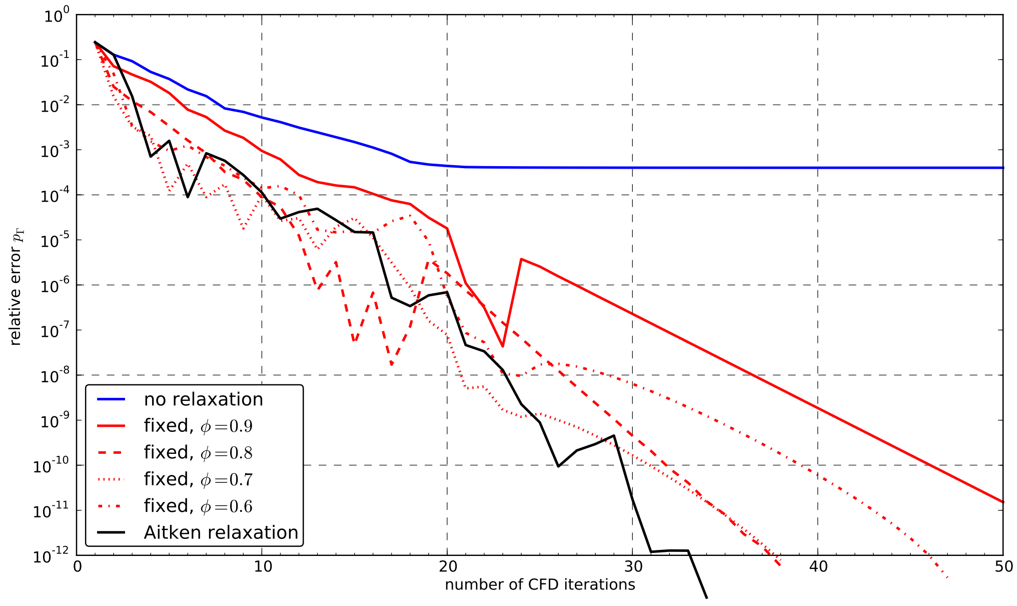


**Figure 6.6.:** Comparison of different relaxation strategies for the loose coupling

To compare the relaxation strategies, Figure 6.6 shows the convergence rate for relaxation on $\mathbf{\Delta p}$ for several fixed $\phi$ and for the Aitken method. All relaxation strategies clearly improve on the non-relaxed procedure. Aitken relaxation is slightly better than the best fixed $\phi$ found, but a significant improvement in case the a priori choice of the fixed $\phi$ was less suitable (e.g. $\phi = 0.9$). This, together with the fact that Aitken relaxation is appropriate for a wider range of simplified model parameters makes it the best choice of relaxation strategy.

## 6.2. A basic autoregressive model

In this section the convergence properties of the loose coupling procedure will be analysed. These properties will be exploited by using a very simple autoregressive model to improve the convergence of the coupled procedure.

### 6.2.1. An alternative expression for the convergence criterion

To see how the convergence of the loose coupling can be further improved, it is instructive to look at the criterion for convergence. The outer loop is said to be converged if $|p_\Gamma^n - p_{\text{target}}| < \epsilon$. Here $p_\Gamma^n$ denotes the average of the vector $\mathbf{p}_\Gamma^n$ with the fluid pressure on the piston for one period.

(In Algorithm 2 this criterion is defined in terms of the force $F$ on the piston, but the pressure can be used equivalently.)

The inner $\mathcal{S}$–$\mathcal{T}$ loop iterates until $p_s \to p_{\text{target}}$ to within some $\epsilon$. As the $\mathcal{S}$–$\mathcal{T}$ iterations are relatively cheap, $p_s$ can be made sufficiently close to $p_{\text{target}}$ on the scale of the error in the outer loop $|p_\Gamma^n - p_{\text{target}}|$, that it can be said that when the inner loop is converged $p_s = p_{\text{target}}$ holds. This means that the convergence criterion of the outer loop can be written as $|P^n| < \epsilon$ with

$$P^n = p_\Gamma^n - p_s^n.$$

Here $p_s^n$ is the pressure on the structure resulting from the converged solution of the inner $\mathcal{S}$–$\mathcal{T}$ using the simple model, in the $n^{\text{th}}$ outer iteration and can be written as

$$p_s^n = \hat{p}_\Gamma^n + \Delta p^n.$$

Using the same notation as in Section 4.4.2, $\hat{p}_\Gamma^n$ denotes the pressure on the piston as calculated by the simple model. The superscript $n$ is added to make the outer iteration explicit. The correction term $\Delta p^n$ is the difference between the CFD solution and the simple model solution in the previous iteration,

$$\Delta p^n = p_\Gamma^{n-1} - \hat{p}_\Gamma^{n-1}. \tag{6.3}$$

Rewriting (6.3) to include $P^n$ gives a recursive formula for $\Delta p^n$. This is the scalar equivalent of Equation (6.1).

$$\begin{aligned}\Delta p^{n+1} &= p_\Gamma^n - \hat{p}_\Gamma^n \\ &= P^n + p_s^n - \hat{p}_\Gamma^n \\ &= P^n + \Delta p^n\end{aligned}$$

So the quantity $P^n$ is exactly the increment of the average of $\mathbf{\Delta p}$ in the $n^{\text{th}}$ outer iteration. This means that convergence of the whole coupling is equivalent to convergence of $\Delta p^n$ to some limit.

Let the limit of the sequence $\Delta p^n$ be denoted $\overline{\Delta p}$. The central idea that forms the basis for further improvements of the loose coupling is that convergence can be improved by adjusting $\mathbf{\Delta p}^{n+1}$ after each outer iteration, such that its average is equal to $\overline{\Delta p}$. As the limit $\overline{\Delta p}$ is unknown, it must be estimated based on previous coupling iterations.

### 6.2.2. Estimation of $\overline{\Delta p}$

The increments $P^n$ of $\Delta p^n$ can be seen as the jumps at every CFD iteration, for example in Figure 5.6(b). The fairly linear behaviour of the relative error on the logarithmic scale suggests that $P^n$ can be accurately modelled by a simple autoregressive model,

$$P^{n+1} = \phi\, P^n.$$

Figure 6.7 shows that a simple geometric sequence is indeed a reasonable approximation of the sequence $P^n$. In this example $P^n$ from an unaltered loose coupling is compared to the geometric sequence starting at $P^2$ and continuing with a common ratio of $\phi = -0.65$. The estimate follows $P^n$ pretty well until about $n = 12$, where the vanishing of $P^n$ starts to slow down. But as $P^n$ has already decreased almost two orders of magnitude, the resulting estimate of $\overline{\Delta p}$ will still be accurate enough to be a useful estimate.

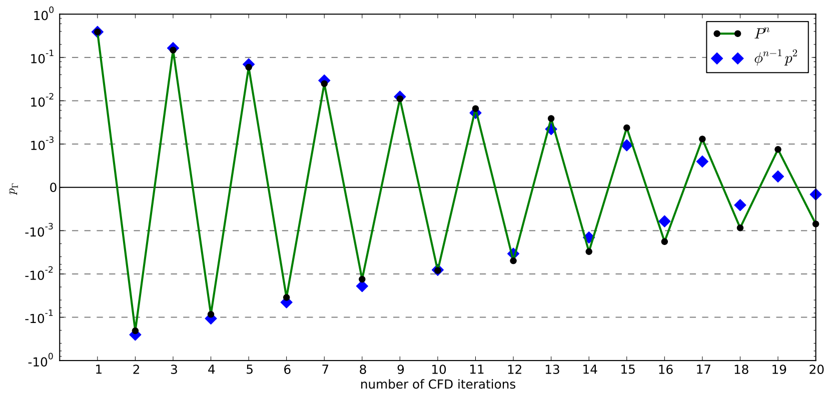


**Figure 6.7.:** Comparison of relative error to a geometric sequence with $\phi = -0.65$

The autoregressive model can be used to find the estimate $\widetilde{\Delta p}$ of the limit $\overline{\Delta p}$. In the case of this simple first order AR model the limit can be calculated explicitly; At the end of iteration $n$ (when $\mathbf{\Delta p}^{n+1}$ is known) the estimate is

$$\begin{aligned}\widetilde{\Delta p} &= \Delta p^{n+1} + \sum_{i=1}^{\infty} \phi^i P^n \\ &= \Delta p^{n+1} + \frac{\phi}{1-\phi} P^n, \qquad \text{for } -1 < \phi < 1.\end{aligned}$$

### Estimation of AR parameter $\phi$

The simplest way of estimating $\phi$ is by using the ratio of the last two iterations.

$$\phi = P^n / P^{n-1} \tag{6.4}$$

### 6.2.3. Adjustment of $\mathbf{\Delta p}$

With the estimate of $\overline{\Delta p}$ known, the next step is to adjust the vector $\mathbf{\Delta p}$ such that its average is equal to $\widetilde{\Delta p}$. The vector $\mathbf{\Delta p}$ is defined element-wise as

$$\mathbf{\Delta p}^{n+1}(j) = \mathbf{p}_\Gamma^n(j) - \hat{\mathbf{p}}_\Gamma^n(j) \qquad , \quad j = 0, \dots, N_t. \tag{6.5}$$

Two different methods for adjusting $\mathbf{\Delta p}$ are considered. The first is to add a constant term to every pressure in $\mathbf{\Delta p}$.

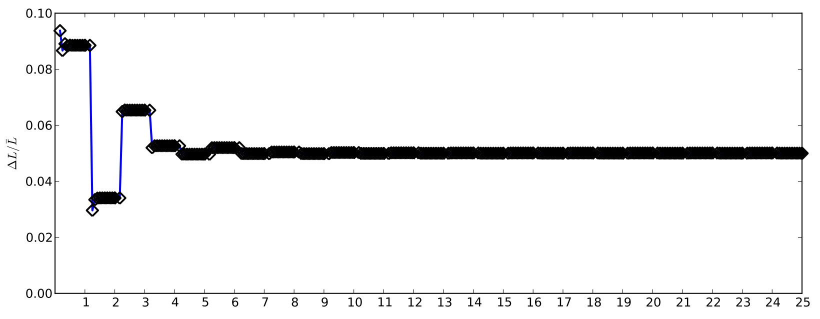
$$\mathbf{\Delta p}^{n+1}(j) = \Big(\mathbf{p}_\Gamma^n(j) - \hat{\mathbf{p}}_\Gamma^n(j)\Big) + \Big(\widetilde{\Delta p} - (p_\Gamma^n - \hat{p}_\Gamma^n)\Big) \quad , \quad j = 0, \dots, N_t \tag{6.6}$$

The second is to multiply every pressure in $\mathbf{\Delta p}$ with a constant term.

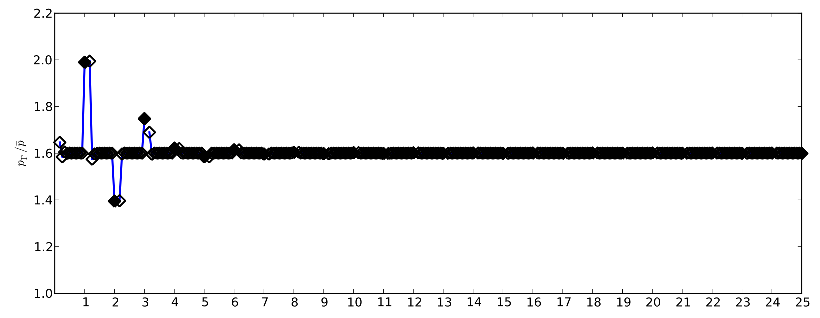
$$\mathbf{\Delta p}^{n+1}(j) = \Big(\mathbf{p}_\Gamma^n(j) - \hat{\mathbf{p}}_\Gamma^n(j)\Big) \frac{\widetilde{\Delta p}}{p_\Gamma^n - \hat{p}_\Gamma^n} \qquad , \quad j = 0, \dots, N_t \tag{6.7}$$

Both methods result in a $\mathbf{\Delta p}$ that has the average of $\widetilde{\Delta p}$. The multiplicative version (6.7) has the advantage that it maintains the overall shape of $\mathbf{\Delta p}$. In particular the zero crossings stay at the same place in the vector, whereas with the additive adjustment from (6.6) the zero crossings of $\mathbf{\Delta p}$ shift. Testing revealed however that in practice the additive version yields better results. Convergence was consistently slower with the multiplicative adjustment. In the following examples thus only the additive adjustment will be used.

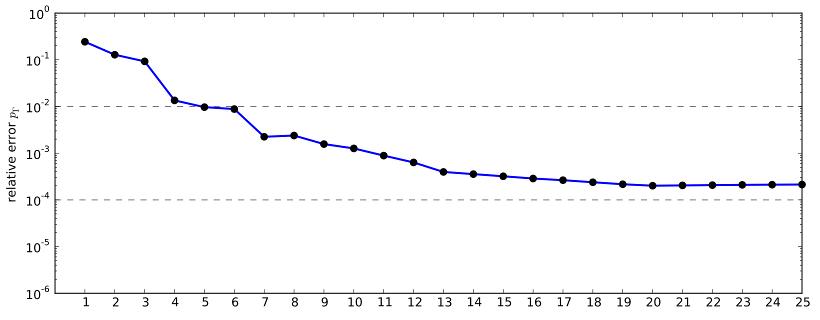
### 6.2.4. Application of the adjustment



(a) Evolution of the trim parameter $\Delta L$ throughout the iterations



(b) One period average $p_\Gamma$. Solid diamonds are the CFD steps.

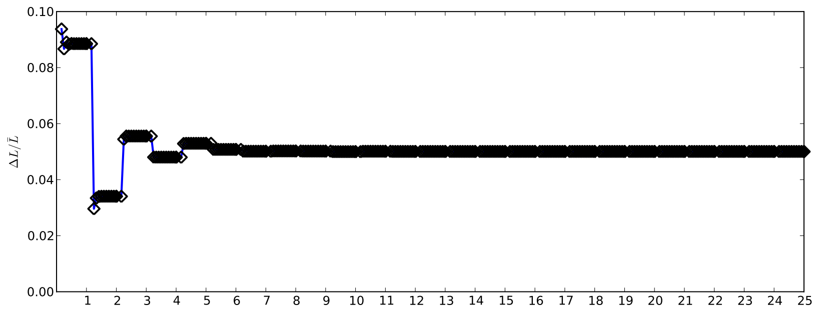


(c) Relative error of $p_\Gamma$ after each CFD step, on a logarithmic scale

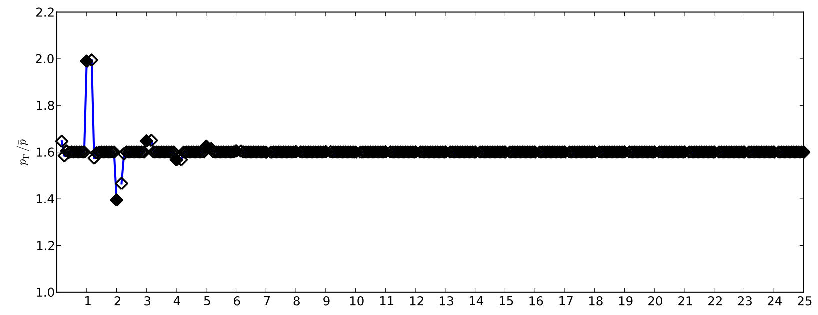
**Figure 6.8.:** 25 iterations of the loosely coupled procedure with a $\mathbf{\Delta p}$ correction only after the third iteration

In Figure 6.8 it is shown how the convergence of the loose coupling is improved when $\mathbf{\Delta p}$ is adjusted. After the third iteration $\phi$ is estimated using (6.4) to be $-0.73$. Using this estimate Equation (6.6) is used to calculate $\mathbf{\Delta p}$ to be used in the fourth iteration. Immediately a big decrease in the relative error of about one order of magnitude can be seen. Compared to the unaltered loose coupling (Figure 5.6) the relative error after 25 CFD steps is somewhat smaller.

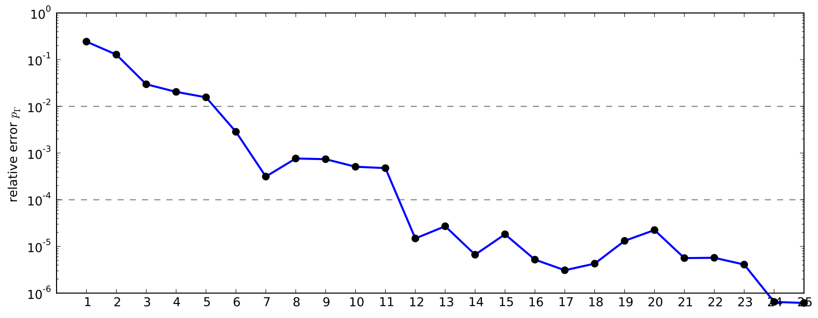
For a better convergence the adjustment of $\mathbf{\Delta p}$ should happen more often. After adjusting $\mathbf{\Delta p}$, however, the $P^n$ that results does not follow the expected pattern anymore. If the estimate $\widetilde{\Delta p}$ is good, the next $P^n$ will be much smaller than expected, as was observed in Figure 6.8. Because the way of estimating $\phi$, (6.4), is only based on the last two iterations, applying the



(a) Evolution of the trim parameter $\Delta L$ throughout the iterations



(b) One period average $p_\Gamma$. Solid diamonds are the CFD steps.



(c) Relative error of $p_\Gamma$ after each CFD step, on a logarithmic scale

**Figure 6.9.:** 25 iterations of the loosely coupled procedure with $\mathbf{\Delta p}$ corrections every 3 iterations

adjustment after every CFD iteration leads to an unstable procedure.

Figure 6.9 shows the result of adjusting $\mathbf{\Delta p}$ every third iteration. Equation (6.6) is applied after CFD steps $2, 5, 8, \ldots$ and for the other iterations the normal updating procedure (6.5) is used. This allows the relative error to settle again in a geometric sequence before another adjustment is made. There is clearly a large decrease in the relative error when the adjustment is applied compared to the unaltered loose coupling. Although convergence is more erratic and the relative error does not reduce monotonically, the prediction of $\overline{\Delta p}$ is good enough to attain an overall better convergence rate over the first 25 iterations.

### 6.2.5. Summary of the results

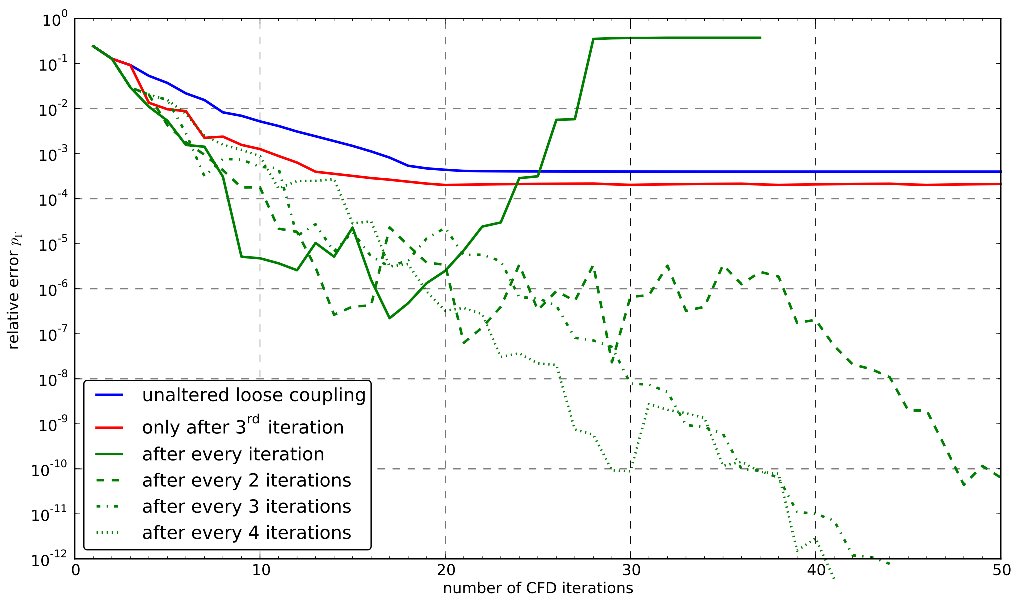


**Figure 6.10.:** Comparison of several strategies using the estimate $\widetilde{\Delta p}$ after different CFD iterations

Figure 6.10 shows the result of suggested improvements of the loose coupling. The convergence to the target $p_\Gamma$ is shown for 50 iterations of the loose coupling. It can be seen that the adjustment of $\mathbf{\Delta p}$ only after the third iteration results in a better convergence for the first 15 iterations, but then also hits a lower bound on the relative error, just like the unaltered loose coupling. The strategy of adjusting $\mathbf{\Delta p}$ every iteration indeed results in a diverging procedure. Updating every second iteration or less results in procedures that are a significant improvement over the loose coupling. The best results are obtained with an update after every fourth iteration. This strategy is comparable in convergence rate to the Aitken relaxation from Figure 6.6 in the first 30 iterations.

## 6.3. Improving the AR model

The simple autoregressive model from the previous section was already a good improvement on the loose coupling. In this section it is shown that the application of the basic time series analysis introduced in Chapter 3 leads to further improvements in the convergence of the coupled system.

### 6.3.1. Using more of the history of $P^n$

Instead of using only the last two observations for the estimation of $\phi$, the complete observed history of $P^n$ can be used. The future development of the average $P^n$ is estimated using an AR(1) model, as explained in Section 3.1. Instead of the simple estimate for the parameter $\phi$ from Equation (6.4), $\phi$ is now chosen such that the total squared error $\sigma^2$ is minimised.

$$\sigma^2 = \sum_{i=1}^{n} \left| P^i - \phi P^{i-1} \right|^2$$

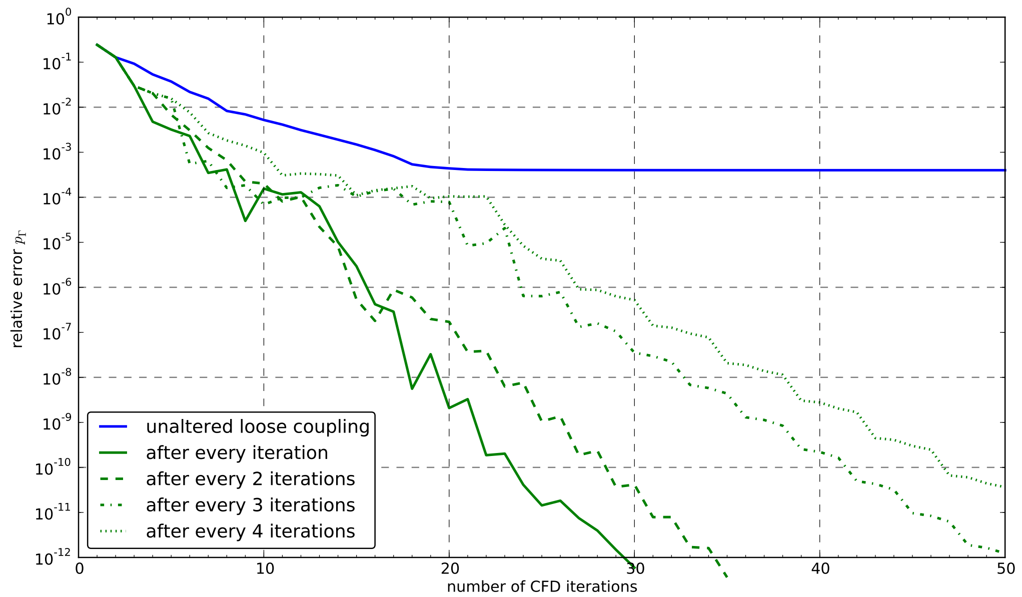


**Figure 6.11.:** Convergence of using the complete history of $P^n$

Figure 6.11 shows that this approach yields a procedure that converges significantly better than the procedure using only the last two observations. The greatest improvement in convergence rate comes from the fact that it is now possible to apply the adjustment to $\mathbf{\Delta p}$ every outer iteration of the loose coupling. With the previous approach using Equation (6.4), adjusting every iteration resulted in an unstable procedure. By using the complete history of $P^n$, the estimates for $\phi$ converge to a single value instead of having a different $\phi$ estimate every iteration. Because of this, it does not matter much for the estimate of $\phi$ whether $P^n$ was adjusted in the previous iteration or not.

### 6.3.2. Using the whole vector $\mathbf{\Delta p}$

In the previous section $P^n$, the increment of the average of $\mathbf{\Delta p}$, was used to estimate $\phi$. A possible enhancement in the estimation of $\phi$ can be to discard the simplification of only using the averages and use the whole vector instead. Figure 6.12 shows how the increments of the full vector $\mathbf{\Delta p}$ evolve during the outer iterations. It can be seen that the whole vector also shows the same oscillatory behaviour during the coupling iterations as the average did (Figure 6.7). This suggests that it is possible to estimate the whole vector with an autoregressive model. In

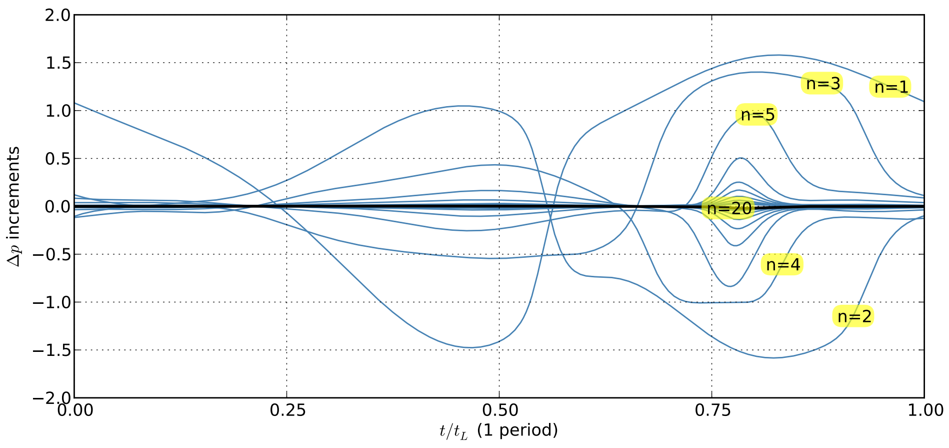


**Figure 6.12.:** Incremental updates for $\mathbf{\Delta p}$ for 20 coupling iterations

this model each increment $\mathbf{P}^n$ of $\mathbf{\Delta p}$ is a scaled version of the previous increment.

$$\begin{aligned} \mathbf{P}^n &= \mathbf{\Delta p}^{n+1} - \mathbf{\Delta p}^n \\ \widetilde{\mathbf{P}}^{n+1} &= \phi\ \mathbf{P}^n \end{aligned} \tag{6.8}$$

Note that $\phi$ is still a scalar here, so the above does not constitute a VAR model as described in Section 3.2. In particular, the forecast of $\mathbf{P}^n$ is still based on the additive update rule of Equation 6.6.

The value for the parameter $\phi$ at iteration $n$ is solved in a least square sense.

$$\phi^n = \arg\min_{\phi} \sum_{i=1}^{n} \sum_{j} \left| \mathbf{P}^i(j) - \phi\ \mathbf{P}^{i-1}(j) \right|^2$$

Figure 6.13 shows that this approach yields a procedure that converges somewhat better than the procedure using the average values.

### 6.3.3. Using a higher order AR($p$) model

Until now the autoregressive models only expressed the relation between two consecutive elements of the series $\mathbf{P}^n$. A higher order autoregressive model, i.e. an AR($p$) model with $p > 1$, can be used to model the relation between several lagging terms of the series. For example, an AR(2) model can be defined as

$$\mathbf{P}^n = \phi_1\ \mathbf{P}^{n-1} + \phi_2\ \mathbf{P}^{n-2}.$$

Tests using a second order AR model to fit the $\mathbf{P}^n$ series did not lead to better predictions and hence no better convergence was obtained. This is most likely due to the fact that a first order model already describes the series very good. Adding more degrees of freedom by using a second order model can then only have a detrimental effect on prediction accuracy. In further sections only first order models will be handled.

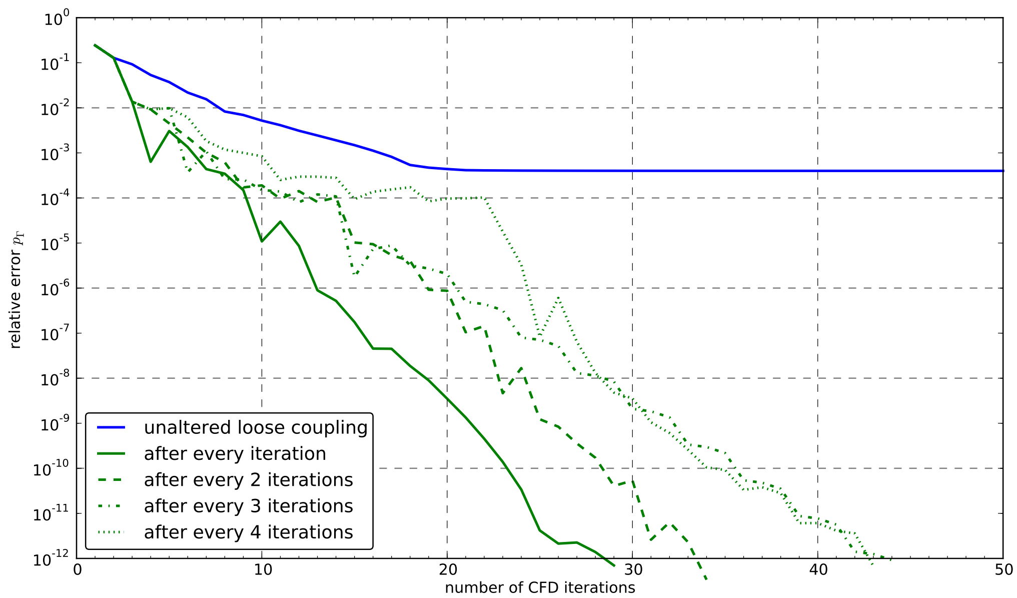


**Figure 6.13.:** Convergence of using the whole vector $\mathbf{\Delta p}$ for estimating the autoregression parameter

### 6.3.4. A simple VAR approach

The model from Section 6.3.2 can be seen as a VAR(1) model by rewriting it in matrix notation.

$$\mathbf{P}^n = \mathbf{\Phi}\ \mathbf{P}^{n-1} \tag{6.9}$$

$$\text{with } \mathbf{\Phi} = \begin{bmatrix} \phi & & & \\ & \phi & & \\ & & \ddots & \\ & & & \phi \end{bmatrix}$$

This can be seen as a subset VAR model, with the non-diagonal entries restricted to zero and additionally, the diagonal entries of the parameter matrix restricted to have the same value.

The difference of this VAR(1) model with the model of Equation (6.8) lies in how the estimate of the future development of $\mathbf{P}^n$ is used to update $\mathbf{\Delta p}^{n+1}$ for the next outer iteration. Instead of first predicting the scalar $\widetilde{\Delta p}$ and then updating $\mathbf{\Delta p}^{n+1}$ such that its average is equal to $\widetilde{\Delta p}$, here the VAR approach is used. In the next outer iteration $\widetilde{\mathbf{\Delta p}}$ is used instead of $\mathbf{\Delta p}^{n+1}$.

$$\widetilde{\mathbf{\Delta p}} = \mathbf{\Delta p}^{n+1} + \sum_{i=1}^{\infty} \tilde{\mathbf{P}}^i$$
$$\text{with } \tilde{\mathbf{P}}^i = \mathbf{\Phi}^i \mathbf{P}^n$$

Figure 6.14 shows that this direct translation of the previous approach into a VAR model results in significantly worse convergence behaviour of the loose coupling. This simple VAR approach is, however, a good basis to start exploring similar models with the parameter matrix $\mathbf{\Phi}$ having another structure.

### 6.3.5. A subset VAR approach using only diagonal entries

A natural extension of the previous approach is to allow only diagonal entries in $\mathbf{\Phi}$, but not to require them to have the same value. Essentially, each element of $\mathbf{P}^n$ has its own scalar

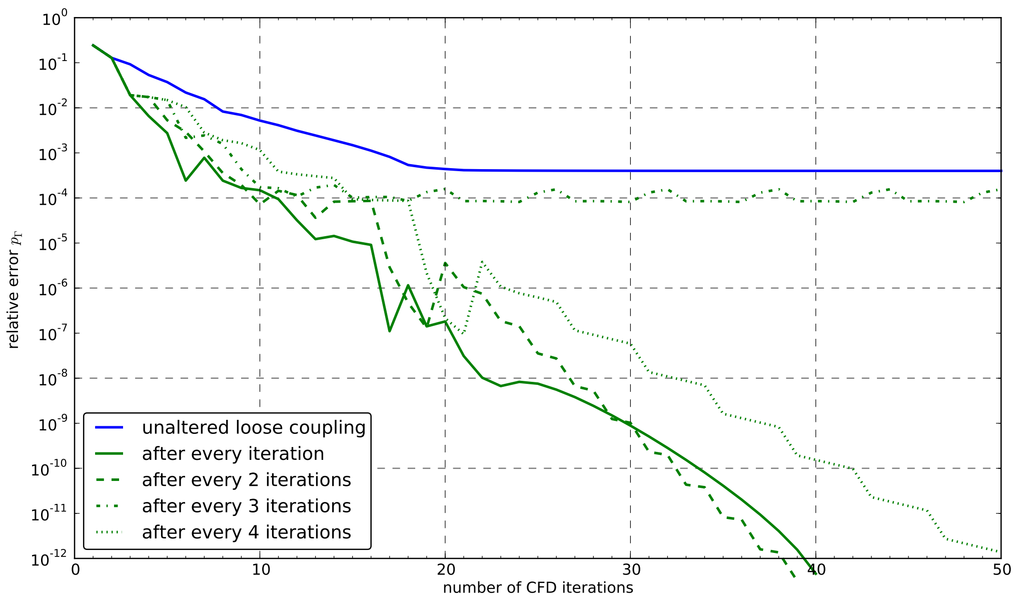


**Figure 6.14.:** Convergence of the loose coupling using a simple VAR model

autoregressive model. The matrix $\mathbf{\Phi}$ in Equation (6.9) then becomes

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1 & & & \\ & \phi_2 & & \\ & & \ddots & \\ & & & \phi_{N_t} \end{bmatrix}$$

$$\text{with} \quad \phi_j = \arg\min_{\phi} \sum_{i=1}^{n} \left| \mathbf{P}^i(j) - \phi\ \mathbf{P}^{i-1}(j) \right|^2 .$$

This does raise some stability concerns, however. Although on average the vector $\mathbf{P}^n$ is decreasing in absolute value, the individual values are not. In Figure 6.12 it can be seen that at some places in the vector, for example at the zero crossings, the next iteration has a larger value. The corresponding autoregressive parameter $\phi$, one of the diagonal entries of $\mathbf{\Phi}$, will then be estimated with an absolute value larger than 1. As a result, $\widetilde{\mathbf{\Delta p}}$ cannot be estimated, because not all the eigenvalues of $\mathbf{\Phi}$ lie within the unit circle. This is prevented by the additional requirement that every diagonal element of $\mathbf{\Phi}$ lies inside the range $[-0.95, 0.95]$.

The result of using this approach is shown in Figure 6.15. It performs about the same as the approach from the previous section, i.e. a bit worse than the previously found best method.

## 6.3.6. A VAR approach using the average as reference

Instead of comparing each element of $\mathbf{P}^n$ to its own value in the previous iteration, it can also be compared to the average of the vector $\mathbf{P}^n$ in the previous iteration. The matrix $\mathbf{\Phi}$ then consists of all identical columns, with each column representing the best fit for the shape of the vector,

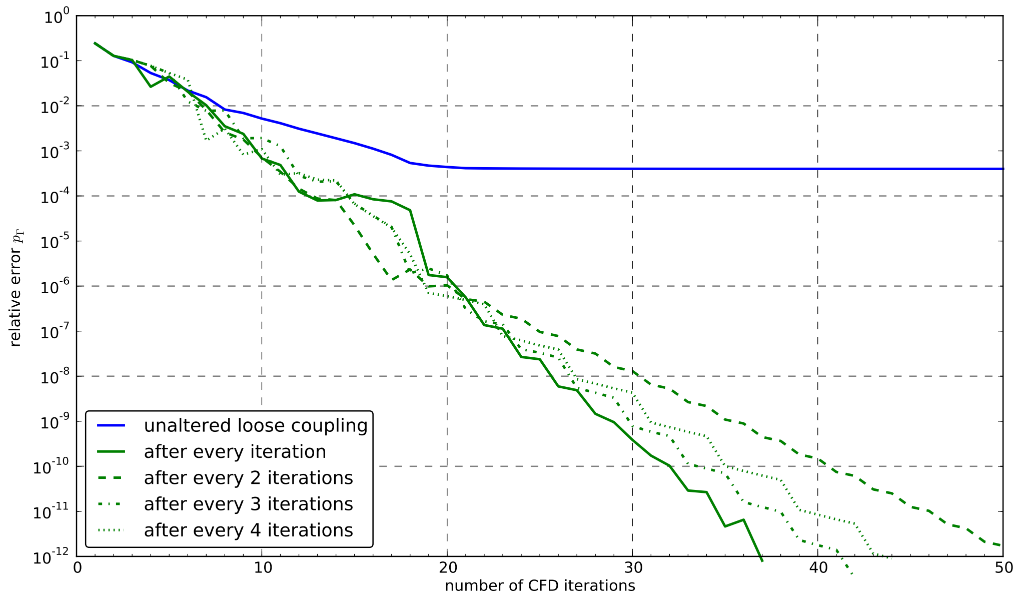


**Figure 6.15.:** Convergence of the loose coupling using a diagonal VAR model

scaled with the average of the previous iteration.

$$\mathbf{\Phi} = \frac{1}{N_t} \begin{bmatrix} \phi_1 & \phi_1 & \cdots & \phi_1 \\ \phi_2 & \phi_2 & \cdots & \phi_2 \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N_t} & \phi_{N_t} & \cdots & \phi_{N_t} \end{bmatrix} \tag{6.10}$$

$$\text{with} \quad \phi_j = \arg\min_{\phi} \sum_{i=1}^{n} \left| \mathbf{P}^i(j) - \phi\ \bar{\mathbf{P}}^{i-1} \right|^2 \tag{6.11}$$

The matrix $\mathbf{\Phi}$ will always have small eigenvalues, because of the $\frac{1}{N_t}$ term.

The result of using this approach is shown in Figure 6.16. It can be seen that this approach achieves good convergence when the adjustment is applied after every iteration of the loose coupling.

### 6.3.7. Including more nonzero elements

When the matrix $\mathbf{\Phi}$ is fitted using the average as reference, as in the previous section, the autoregressive model is of course not a perfect fit for the sequence of vectors $\mathbf{P}^n$ resulting from the loose coupling iterations. There is still a residual error $\tilde{\sigma}^2$. For row $j$ of the system of equations this residual is

$$\tilde{\sigma}_j^2 = \sum_{i=1}^{n} \left| \mathbf{P}^i(j) - \phi_j \bar{\mathbf{P}}^{i-1} \right|^2 ,$$

which in matrix notation can be written as

$$\tilde{\sigma}_j^2 = \sum_{i=1}^{n} \left| \left[ \mathbf{P}^i - \mathbf{\Phi} \mathbf{P}^{i-1} \right] (j) \right|^2 .$$

The proposed approach in this section is to allow an element in every row to be adjusted freely from the other elements in order to minimise the residual error. Which element in each row is

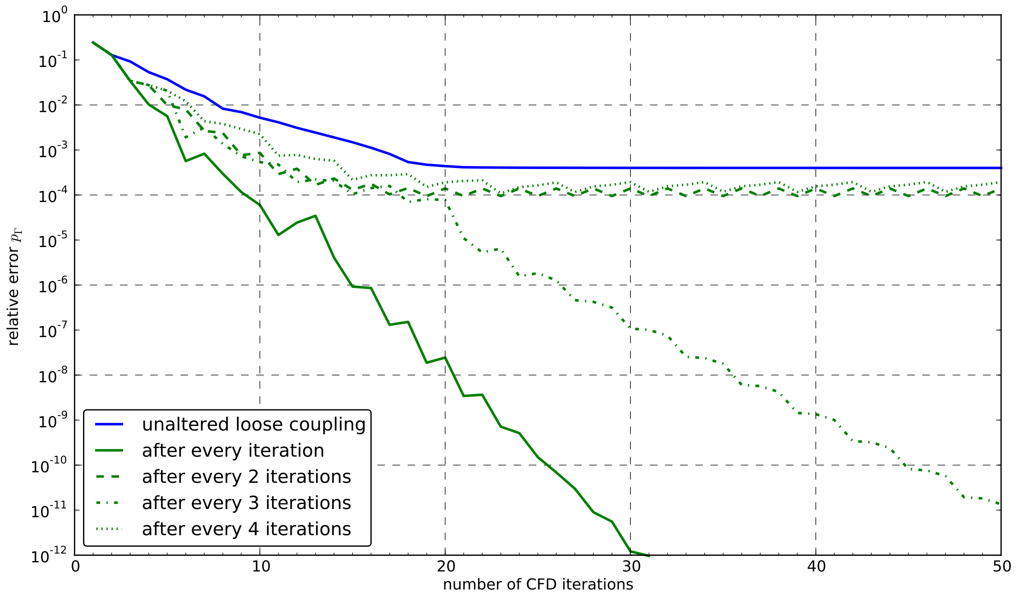


**Figure 6.16.:** Convergence of the loose coupling using an average-based VAR model

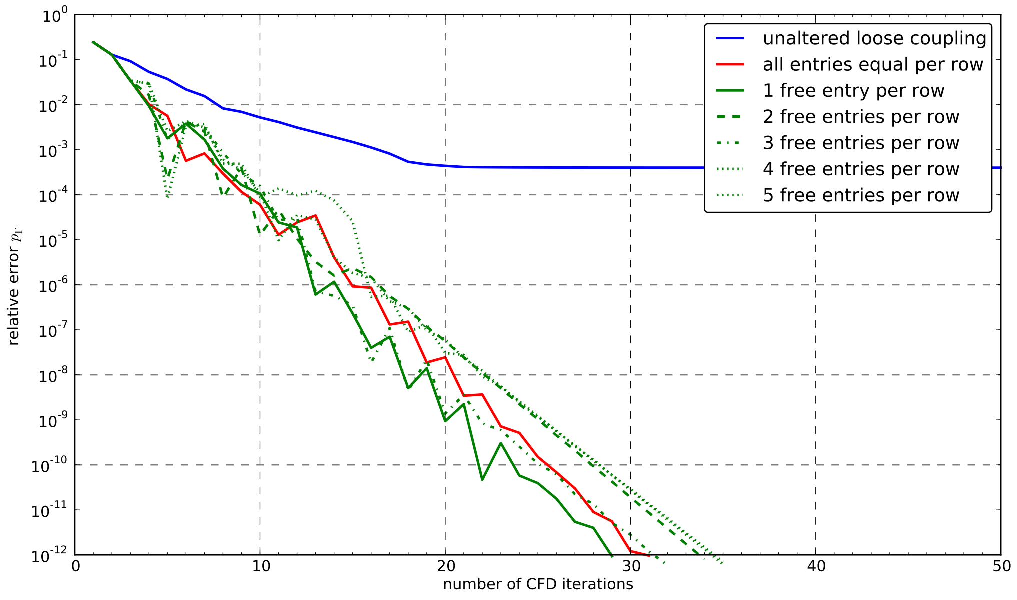


**Figure 6.17.:** Convergence of the loose coupling using a VAR model with limited degrees of freedom per row

adjusted is not given a priori, as for example was the case with the diagonal $\mathbf{\Phi}$, but is determined by choosing the element that decreases the residual error the most. This can be done multiple times, each time selecting an additional element in a row that is allowed to have a different value than the value found by using the average as reference.

The result of using this approach is shown in Figure 6.17. For comparison the method of the previous section is shown in red. It can be seen that adding one entry per row gives better convergence, but any subsequent increase of the number of nonzero elements in $\mathbf{\Phi}$ only decreases convergence. It can thus be concluded that two different values per row give the autoregressive model enough degrees of freedom to fit the series $\mathbf{P}^n$ accurately.

### 6.3.8. An AIC-based subset VAR approach

The last possible improvement to the prediction of $\overline{\mathbf{\Delta p}}$ is to use the subset VAR technique with the Akaike Information Criterion as introduced in Section 3.2. The difference with the approach from the previous section is that the first estimate for the parameters is not necessarily based on the average $P$. Furthermore, the number of unique (nonzero) elements in each row of $\mathbf{\Phi}$ is not fixed, but is determined by using the AIC to find the best number of parameters.

Using the average as reference yields a more stable procedure than only predicting based on individual elements of the previous $\mathbf{P}$. However, the AIC gives a penalty for every nonzero element in $\mathbf{\Phi}$, which means that a matrix like (6.10) will not have a low AIC. To overcome this problem, the procedure for finding the parameter matrix with the lowest AIC is used on a modified matrix $\widehat{\mathbf{\Phi}}$, an $N_t \times N_t + 1$ matrix where the last column contains the autoregressive coefficients with the average as reference. The vector $\widehat{\mathbf{P}}$ is the vector $\mathbf{P}$ augmented with its average $P$,

$$\widehat{\mathbf{P}}^n = \left[\mathbf{P}^{\mathbf{n}} \middle| P^n\right] = \mathbf{A}\mathbf{P}^{\mathbf{n}},$$

$$\text{where} \quad \mathbf{A} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \\ \frac{1}{N_t} & \cdots & \frac{1}{N_t} \end{bmatrix}.$$

After $n$ outer iterations of the loose coupling the new estimate $\widetilde{\mathbf{\Delta p}}$ of the pressure difference is

$$\widetilde{\mathbf{\Delta p}} = \mathbf{\Delta P}^n + \sum_{i=0}^{\infty} \left(\widehat{\mathbf{\Phi}}\mathbf{A}\right)^i \mathbf{P}^n,$$

with $\widehat{\mathbf{\Phi}}$ the minimiser of

$$\text{AIC} = \ln \tilde{\sigma}^2 + \frac{2}{n} n_{\text{param}}$$

$$\text{where} \quad \tilde{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} \left| \mathbf{P}^i - \widehat{\mathbf{\Phi}}\widehat{\mathbf{P}}^{i-1} \right|^2,$$

and $n_{\text{param}}$ the number of nonzero elements in $\widehat{\mathbf{\Phi}}$ under the condition that the eigenvalues of $\widehat{\mathbf{\Phi}}\mathbf{A}$ lie within the unit circle. The heuristic used for finding the AIC-minimising $\widehat{\mathbf{\Phi}}$ is the top-down strategy from Section 3.2.1.

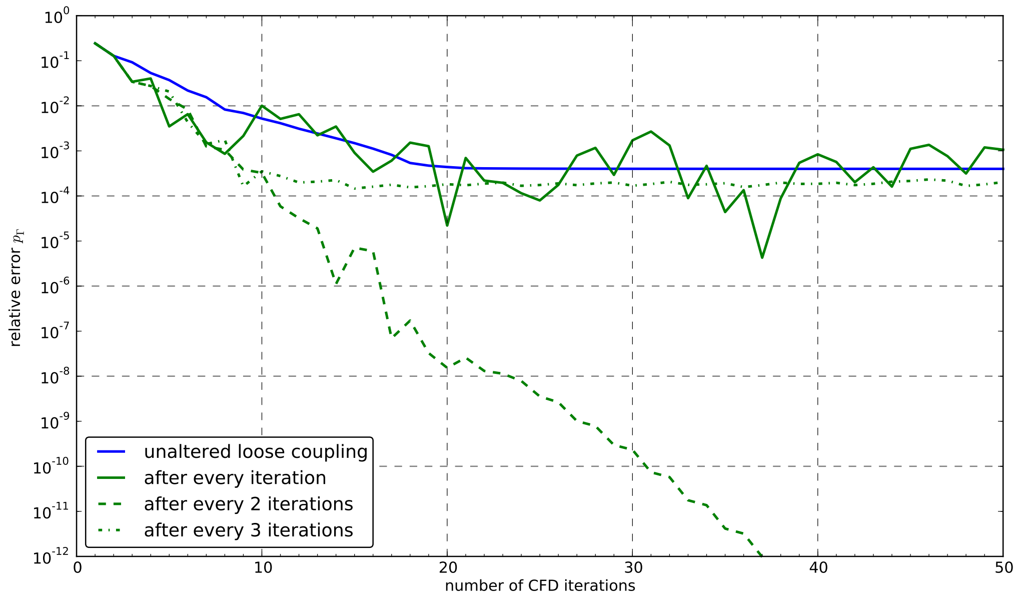


**Figure 6.18.:** Convergence of the loose coupling using an AIC-based subset VAR model

The results of this subset VAR approach are shown in Figure 6.18. Convergence is improved only when the adjustment of $\mathbf{\Delta p}$ applied every other iteration. This is quite a step back from the previous best result.

## 6.4. Comparison of autoregressive techniques

Figure 6.19 shows the best results from the autoregressive models of Sections 6.2, 6.3.2 and 6.3.7 compared against the original loose coupling and the procedure using Aitken relaxation. The basic AR model performs slightly worse then Aitken relaxation. The improved scalar AR model and the constrained VAR model have the best convergence.

The rate of convergence is measured in the number of CFD iterations required to get $p_\Gamma$ close to $p_{\text{target}}$ to within a certain error. The idea here is that the number of CFD iterations is a reasonable proxy for the amount of computational work involved. In comparison with the CFD model the other physical models involved (the simple CFD model, the CSD model and the trim model) indeed all require significantly less work, both for this model problem and for an actual rotorcraft simulation. The adjustment of $\mathbf{\Delta p}$ using relaxation or an AR model is computationally trivial. However, for the VAR methods of Sections 6.3.7 and 6.3.8 the overhead of the adjustment between the loose coupling outer iterations can become quite significant. This is because in order to find the element of $\mathbf{\Phi}$ that gives the largest reduction of the AIC a linear least-squares problem has to be solved for each of the approximately ${N_t}^2$ candidate elements. For the model problem with the parameter values from Table A.1 ($N = 400$, $N_t = 300$) this resulted in the wall clock time doubling from one hour to two hours compared to the unaltered loose coupling.

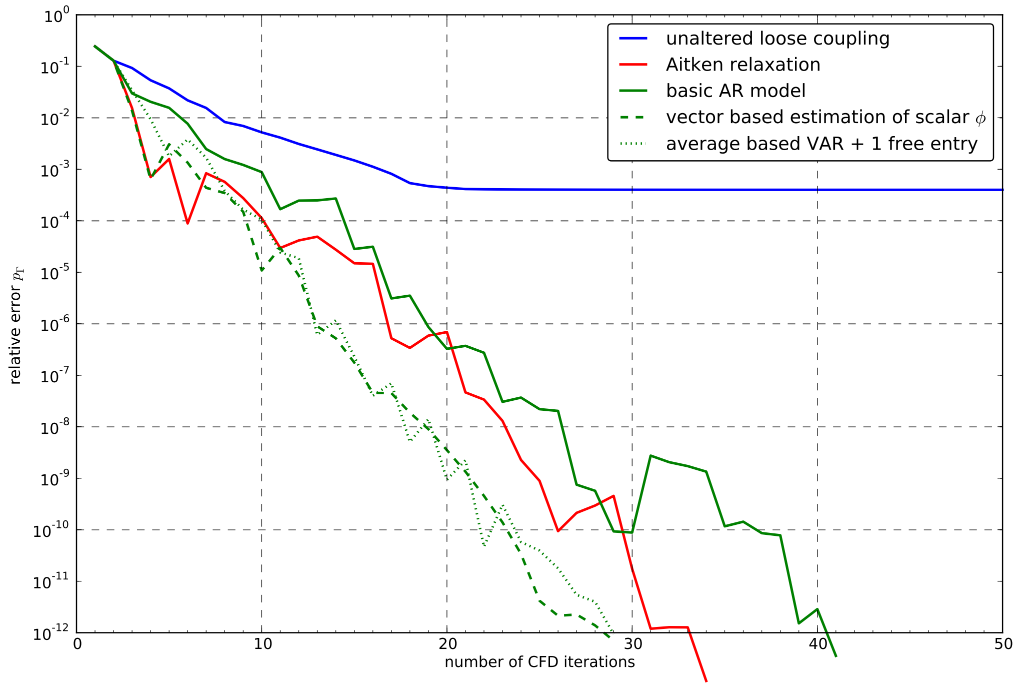


**Figure 6.19.:** Comparison of convergence using Aitken relaxation and several AR models

CHAPTER 7

# Conclusions

## Interaction laws

In Chapter 2 several examples were shown where the use of an interaction law improves the convergence of a partitioned iterative procedure modelling a coupled system. The concept of an interaction law can also be applied to the three models involved in simulating trimmed rotorcraft flight, resulting in the well-known loose coupling approach.

The use of an autoregressive model to predict the delta loads in further iterations can be seen as another interaction law. So the first interaction law uses the simple model as an approximation to the full CFD model and the second interaction law extrapolates the discrepancy between those models in order to find an even better estimate of the final CFD solution.

The formulation of the loose coupling as an interaction law can give more insight as to why this approach works. Further study is needed to see what conditions the simple model must satisfy in order to actually improve the convergence of the loose coupling and whether the use of the autoregressive models influences that. This could be done by studying the effect of the interaction law on the eigenvalues of the partitioned system, as was done in Section 2.3.

## Model problem

The goal of the model problem was to find a simple coupled CFD-CSD system that still had some resemblance to the rotorcraft physics. That proved to be the case, although of course the real verification of that can only be seen by applying the found improvements to a real rotorcraft simulation.

A possible improvement to the coupling approach of the model problem that has not been researched is to run the CFD model for fewer periods between each CSD-simple model iteration. This has the effect that the trim is being adjusted more often, before the CFD model has reached a periodic solution. A variation of this is to make use of the space-time formulation of the CFD model and calculate a periodic solution with increasing accuracy, interspersed with trimming iterations of the CSD-simple model. In that situation both the convergence to a periodic CFD solution and the convergence to a trimmed CSD-coupled solution are combined in each situation, so it would be interesting to see whether an autoregressive model can still be used to improve the convergence.

The Aitken relaxation strategy for the delta loads exchanged between the CFD and CSD models proved to be a simple way to improve convergence. This method achieves good results without any analysis necessary. It assumes very little of the convergence process and is thus more general than the autoregressive approach. Better results can be achieved, however, by carefully modelling the time series.

### Time series analysis

The time series $P^n$ resulting from the loose coupling can be analysed adequately with an autoregressive model. The fitted model can then be used to enhance convergence in subsequent iterations of the loose coupling.

A simple AR model (Section 6.3.2) works the best for the model problem. The fact that neither a higher order AR model, nor a VAR model leads to better results could mean that the convergence of the loose coupling results in a simple series that can be accurately modelled with a simple AR model. This should be verified on more accurate rotorcraft coupled models.

A complicating aspect of the time series analysis for the loose coupling is that the result of the autoregressive modelling is used to adjust the time series itself. This causes the underlying model to change and could lead to a slower convergence of the series. More analysis is needed to study the effect of the adjustments on the series prediction.

### Applicability to rotorcraft modelling

The easiest improvement to add to a rotorcraft coupled procedure is the Aitken relaxation. In the literature it has shown to perform well in a wide variety of situations.

For the autoregressive approach it first has to be checked which type of model best fits the coupling data from a large scale simulation. The list of models presented in Section 6.3 are a good starting point for this analysis, but it can be extended with simple and more advanced VAR techniques.

Furthermore the computational feasibility of extensive autoregressive modelling has to be considered, as the vectors and matrices will be much larger. Each data point in the time series being analysed is a vector with the number of elements equal to the amount of interface nodes between the CSD and CFD models. It is likely that it is not possible to consider as many different subset models as was done in Section 6.3.8 and a simpler model must be chosen.

It is expected, however, that the CFD model will still account for the largest part of the computational work. This means that even when the autoregressive method proposed takes considerable effort to fit to the convergence data, it will still improve the overall run time of the simulation by drastically reducing the number of outer iterations needed to reach the desired accuracy. This can for example make it possible to use more advanced CFD models during the design process of a new rotor blade, when many model runs are necessary for the iterative design process.

# Acknowledgements

I'd like to thank the National Aerospace Laboratory NLR for the opportunity to conduct the research for this Master's Thesis at the NLR. Furthermore, my thanks goes my supervisors Jaap van der Vegt (University of Twente) and Harmen van der Ven (National Aerospace Laboratory) for their guidance and supervision during this final project.

Finally, my gratitude goes to my wife, Hanna. Without her support I could not have finished this thesis.

# Bibliography

[1] C. Michler, E. H. van Brummelen, S. J. Hulshoff, and R. de Borst. The relevance of conservation for stability and accuracy of numerical methods for fluid-structure interaction. *Computer Methods in Applied Mechanics and Engineering*, 192(37-38):4195–4215, 2003.

[2] E. H. van Brummelen, S. J. Hulshoff, and R. de Borst. Energy conservation under incompatibility for fluid-structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 192(25):2727–2748, 2003.

[3] S. Piperno and C. Farhat. Partitioned procedures for the transient solution of coupled aeroelastic problems - part II: energy transfer analysis and three-dimensional applications. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3147–3170, 2001.

[4] A. Datta, M. Nixon, and I. Chopra. Review of rotor loads prediction with the emergence of rotorcraft CFD. *Journal of the American Helicopter Society*, 52(4):287–317, 2007.

[5] J.C. Kok, J. van Muijden, S.S. Burgers, H.S. Dol, and S.P. Spekreijse. Enhancement of aircraft cabin comfort studies by coupling of models for human thermoregulation, internal radiation, and turbulent flows. Technical Paper NLR-TP-2006-391, National Aerospace Laboratory NLR, 2006.

[6] A.E.P. Veldman. Quasi-simultaneous viscous-inviscid interaction for transonic airfoil flow. In *4th AIAA Theoretical Fluid Mechanics Meeting*, June 6–9 2005, Toronto, Ontario Canada, 2005. Paper AIAA 2005-4801.

[7] A.E.P. Veldman. A simple interaction law for viscous–inviscid interaction. *Journal of Engineering Mathematics*, 65(4):367–383, Dec 2009.

[8] E.G.M. Coenen. *Viscous-inviscid interaction with the quasi-simultaneous method for 2D and 3D aerodynamic flow.* PhD thesis, Rijksuniversiteit Groningen, 2001.

[9] R.S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Inc., Englewood Cliffs, NJ, USA, 1962.

[10] G. Servera, P. Beaumier, and M. Costes. A weak coupling method between the dynamics code HOST and the 3D unsteady Euler code WAVES. In *Proceedings of the 26th European Rotorcraft Forum*, September 26–29, 2000, The Hague, 2000.

[11] K. Pahlke and B. van der Wall. Progress in weak fluid-structure-coupling for multibladed rotors in high speed forward flight. In *Proceedings of the 28th European Rotorcraft Forum*, September 17-20, 2002, Bristol, 2002.

[12] G. Kirchgässner and J. Wolters. *Introduction to Modern Time Series Analysis.* Springer, 1st edition, October 2007.

[13] H. Lütkepohl. *New Introduction to Multiple Time Series Analysis.* Springer, 1st edition, March 2005.

[14] J. J. W. van der Vegt and H. van der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. general formulation. *J. Comput. Phys.*, 182(2):546–585, 2002.

[15] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer-Verlag, Berlin, 1999.

[16] G. B. Jacobs, D. A. Kopriva, and F. Mashayek. A conservative isothermal wall boundary condition for the compressible Navier–Stokes equations. *Journal of Scientific Computing*, 30(2):177–192, 2007.

[17] U. Küttler and W. Wall. Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72, Dec 2008.

APPENDIX A

# List of variables and parameters

| | |
|---|---|
| $\bar{L}$ | 96.0 m |
| $\Delta L$ | 9.0 m |
| $A$ | 0.50 $m^2$ |
| $\rho_I$ | 1.25 $kg/m^3$ |
| $u_I$ | 0.0 m/s |
| $p_I$ | 1030 hPa |
| $p_a$ | 3000 hPa |
| $m$ | 24.0 kg |
| $k$ | 120000 N/m |
| $z_I$ | 0.0 m |
| $\dot{z}_I$ | 0.0 m/s |
| $N$ | 400 |
| $t_L$ | 0.26 s |
| $\Delta t$ | 1/300 $t_L$ |
| $C_1$ | 0.72 $\bar{L}$ |
| $C_2$ | 2.18 $p_I$ |

**Table A.1.:** Parameter values used in numerical experiments

| | | | |
|---|---|---|---|
| dimensions | | | |
| $x$ | length along the cylinder | $t$ | time |
| cylinder geometry | | | |
| $l(t)$ | length of the cylinder at $t$ | | |
| $z(t)$ | displacement of the spring | | |
| $L(t)$ | prescribed structure motion | | |
| $\bar{L}, \Delta L$ | mean length and amplitude of prescribed motion | | |
| $t_L, \omega_L$ | period and frequency of prescribed motion | | |
| $A$ | cross-sectional area of the cylinder | | |
| structure parameters | | | |
| $m$ | mass of the piston | $k$ | spring constant of the structure |
| fluid variables defined on $(x,t)$ for $t > 0$ and $0 < x < l(t)$ | | | |
| $\rho$ | mass density | $e$ | total energy |
| $u$ | fluid velocity | $p$ | pressure |
| $U$ | fluid state = $[\rho \ \rho u \ \rho e]^T$ | | |
| initial conditions | | | |
| $\rho_I, u_I, p_I$ | initial fluid state ($U_I$) | | |
| $z_I, \dot{z}_I$ | initial spring displacement and velocity | | |
| $l_I$ | initial length of the cylinder | | |
| variables at fluid–structure interface | | | |
| $p_a$ | pressure from outside on the piston | | |
| $p_s(t)$ | pressure on the piston from inside the cylinder | | |
| $u_\Gamma(t), p_\Gamma(t)$ | fluid velocity and pressure at the piston | | |
| $\hat{p}_\Gamma(t)$ | pressure from the simple model | | |
| discretisation parameters | | | |
| $N$ | number of elements in one time slab | | |
| $\Delta t$ | time step | | |
| $N_t$ | number of time steps in one period | | |
| vectors of length $N_t$ | | | |
| $\mathbf{z}$ | periodic motion of the piston | | |
| $\mathbf{p_s}$ | periodic fluid pressure on the piston | | |
| $\mathbf{p_\Gamma}, \hat{\mathbf{p}}_\mathbf{\Gamma}$ | periodic fluid pressure of the CFD and simple model at the piston | | |
| $\mathbf{\Delta p}$ | difference between CFD and simple model pressure | | |

**Table A.2.:** List of symbols

APPENDIX B

# Model problem Python code

## modelparameters.py

```
from numpy import array, sqrt, pi

resolution = "high"
N, nsteps, residual_tolerance = {'verylow': (15,  15, 1e-8 ),
                                 'medium': ( 60,  60, 1e-9 ),
                                 'high':  ( 400, 300, 1e-9 ),
                                }[resolution]
periodicity_tolerance = 8.0e-3

# Model parameters in SI units
L_D = 96.0        # m      Length of the cylinder
D_D = 0.8         # m      Diameter of the cylinder
rho_I_D = 1.25    # kg/m3  Density in the cylinder
p_I_D = 1030e2    # Pa     Pressure in the cylinder
p_a_D = 3000e2    # Pa     Pressure outside the cylinder
m_D = 24.0        # kg     Mass of the piston
k_D = 120000      # N/m    Spring constant
z_I_D = 0.0       # m      Initial displacement of the spring
v_I_D = 0.0        # m/s   Initial velocity of the spring
DL_D = 9.0          # m    Amplitude of right wall motion
t_L_D = 0.26         # s   Period of prescribed motion
t_end_D = t_L_D*12  # s    Simulation end time
dt_D = t_L_D/nsteps # s    Simulation time step

gamma = 1.4
def p_f(U):
  return (gamma - 1.0) * (U[2] - 0.5 * U[1] * U[1] / U[0])

# Scale factors for nondimensionalization
l_S = L_D
rho_S = rho_I_D
p_S = p_I_D
v_S = sqrt(p_S/rho_S)
t_S = l_S / v_S
m_S = rho_S * l_S**3
k_S = p_S * l_S

# Dimensionless time parameters
dt = dt_D / t_S
t_end = t_end_D / t_S
Nperiod = int(round(t_L_D/dt_D))
omega_L = 2.0 * pi / (dt * Nperiod)

A = pi/4.0 * (D_D/l_S)**2
p_a = p_a_D / p_S * A

# Dimensionless inital fluid state in the cylinder, scaled with the area of the piston
rho_I = rho_I_D / rho_S * A
p_I = p_I_D / p_S * A
U_I = array([rho_I, 0.0, p_I / (gamma-1.0)])

z_I = z_I_D / l_S
v_I = v_I_D / v_S

# Dimensionless parameters
L = L_D / l_S
DL = DL_D / l_S
k = k_D / k_S
m = m_D / m_S
```

## modelvariables.py

```
from numpy import zeros, ones, ceil

def reset_time():
  from modelparameters import N, dt, t_end

  # Reset time variables
  global t, ti
  t = 0.0
  ti = 0

  # Reset variables to save complete time-dependent history
  global l_all, U_all, pGamma_all
  l_all = zeros([1+ceil(t_end/dt)])
  pGamma_all = zeros([1+ceil(t_end/dt)])
  U_all = ones([N, 3, 1+ceil(t_end/dt)])
reset_time()
```

## CFD_approx.py

```
from modelparameters import gamma, dt, p_I

C1 = 0.72
C2 = 2.18

l_I = 0.0

def calculate_pressure(l0, l1):
  l = (l0 + l1) / 2
  v = (l1 - l0) / dt
  length_ratio = (l_I - C1) / (l - C1)

  return p_I * length_ratio ** gamma - v * p_I * C2
```

## CFD_spacetime.py

```
from numpy import sqrt, log10, absolute, maximum, zeros, array, arange, vstack, newaxis
from modelparameters import gamma, p_f, N, dt, Nperiod, residual_tolerance, U_I

RT_inf = p_f(U_I) / U_I[0]

# Calculate flow solution for one time slab
def _timeslab_spacetime(l0, l1, U_t0, U):
  h0 = l0 / N
  h1 = l1 / N

  H_hat = zeros([N+1, 3])

  # Pseudo time stepping
  for n in xrange(2000):
    # Left boundary: add dummy state before calculating numerical flux
    U_xl = U[0,:]
    rho_l = U_xl[0]
    rhou_l = - U_xl[1]
    rhoe_l = - U_xl[2] + 2/(gamma-1.0) * RT_inf * rho_l + rhou_l**2 / rho_l
    U_l_dummy = array([rho_l, rhou_l, rhoe_l])
    U_d = vstack([U_l_dummy, U])

    # Calculate F1
    rho_inv = 1.0 / U_d[:,0]
    u = U_d[:,1] * rho_inv
    a = sqrt(gamma * (gamma-1.0) * (U_d[:,2] * rho_inv - 0.5 * u**2))
    F1 = zeros([N+1,3])
    F1[:,0] = U_d[:,1]
    F1[:,1] = (gamma-1.0) * U_d[:,2] + (3.0-gamma)/2.0 * U_d[:,1]**2 * rho_inv
    F1[:,2] = gamma * U_d[:,2]*u - (gamma-1.0)/2.0 * U_d[:,1] * u**2

    # Estimate largest eigenvalue lambda_hat
    v = arange(0.0, N+1) * (h1-h0) / dt
    lambda_l = absolute(u[0:N]   - v[0:N]) + a[0:N]
    lambda_r = absolute(u[1:N+1] - v[0:N]) + a[1:N+1]
    lambda_hat = maximum(lambda_l, lambda_r)
    lambda_hat_max = max(lambda_hat)

    # Calculate numerical flux
    H_hat[0:N,:] = 0.5 * (F1[0:N,:] + F1[1:N+1,:])                    \
                 - 0.5 * v[0:N,newaxis] * (U_d[0:N,:] + U_d[1:N+1,:])        \
                 + 0.5 * lambda_hat[0:N,newaxis] * (U_d[0:N,:] - U_d[1:N+1,:])

    # Right boundary: use *-state to calculate flux directly
    U_xr = U[N-1,:]
    a_r = sqrt(gamma * (gamma-1.0) * (U_xr[2] - 0.5*U_xr[1]**2/U_xr[0]) / U_xr[0])
    uva_r = (U_xr[1]/U_xr[0] - (l1-l0)/dt) / a_r
    if uva_r <= 0:
      p_star = p_f(U_xr) * (1 + (gamma-1.0)/2 * uva_r) ** (2*gamma/(gamma-1.0))
    else:
      p_star = p_f(U_xr) * (1 + gamma*(gamma+1.0)/4 * uva_r**2 * (1 + sqrt(1+(4/(gamma+1.0)/uva_r)**2)))
    H_hat[N,0:3] = [0, p_star, (l1-l0)/dt * p_star]
    # Calculate residual from fluxes and do pseudo time step
    res = h1 * U - h0 * U_t0 + dt * ( H_hat[1:N+1,:] - H_hat[0:N,:] )
    dtau = 1.0 / ((h0+h1)/2 + dt * lambda_hat_max)
    U = U - dtau * res

    # Stop iterating when required precision is reached
    res_max = absolute(res).max(0)
    if res_max.max() < residual_tolerance:
      break

  #print "\tCFD: %d" % n, "\t(%4.1f  %4.1f  %4.1f)" % tuple(log10(res_max)), " - ",
  return U


def calculate_pressure(l0, l1):
  from modelvariables import ti, U_all

  # Set fluid state in previous time slab
  if ti == 0:
    U_t0 = zeros([N, 3])
    U_t0[:,:] = U_I.copy()
  else:
    U_t0 = U_all[:,:,ti-1]

  # Initial guess for fluid state
  if ti < Nperiod*5:
    if ti < 3:
      U = U_t0.copy()
    else:
      U = 3 * U_all[:,:,ti-1] - 3 * U_all[:,:,ti-2] + U_all[:,:,ti-3]
  else:
    U = U_all[:,:,ti-Nperiod].copy()

  U = _timeslab_spacetime(l0, l1, U_t0, U)
  U_all[:,:,ti] = U

  pGamma = p_f(U[N-1,:])
  return pGamma
```

## FSI_timestepping.py

```
from math import sqrt, sin, cos, pi
from numpy import log10, average, isnan

import CFD_spacetime
import CFD_approx
import modelvariables
from modelparameters import residual_tolerance, periodicity_tolerance, dt, Nperiod, A, omega_L, p_a

def reset_time():
  modelvariables.reset_time()
  from modelparameters import z_I, v_I, L, DL, k, m
  global ti_periodic, z_0, v_0, _omega_km, _m, _k_inv, length, dlengthdtt
  ti_periodic = 0
  z_0 = z_I
  v_0 = v_I
  _omega_km = sqrt(k/m)
  _m = m
  _k_inv = 1.0 / k
  length = lambda t: L + DL * cos(omega_L * t)
  dlengthdtt = lambda t: - DL * omega_L**2 * cos(omega_L * t)
  CFD_approx.l_I = length(0)
reset_time()


# Structural model
def calculate_structure(p_s, dLdtt0, dLdtt1):
  sin_dt = sin(_omega_km*dt)
  cos_dt = cos(_omega_km*dt)
  beta0 = (p_s - p_a - _m * dLdtt0) * _k_inv
  beta1 = _m * (dLdtt0 - dLdtt1) / dt * _k_inv
  z_1 = z_0 * cos_dt + v_0 * sin_dt/_omega_km + beta0 * (1.0 - cos_dt) + beta1 * (dt - sin_dt/_omega_km)
  v_1 = v_0 * cos_dt - z_0 * sin_dt*_omega_km + beta0 * sin_dt*_omega_km + beta1 * (1.0 - cos_dt)
  return z_1, v_1


# Use prescribed l_period
def do_timestep_prescribed_l(l_period, pGamma_period):
  from modelvariables import t, ti
  global ti_periodic

  # Calculate fluid state
  l0 = l_period[ti % Nperiod]
  l1 = l_period[(ti+1) % Nperiod]
  pGamma = CFD_spacetime.calculate_pressure(l0, l1)
  # Update ti_periodic if solution is not yet periodic
  if abs(pGamma - pGamma_period[ti % Nperiod])/pGamma > periodicity_tolerance:
    ti_periodic = ti
  pGamma_period[ti % Nperiod] = pGamma

  # Advance timestep
  modelvariables.ti = ti + 1
  modelvariables.t = t + dt


# Use structural model
def do_timestep(l_period, pGamma_period, pDelta, UseApprox=False):
  from modelvariables import t, ti
  global ti_periodic, z_0, v_0

  L0 = length(t)
  L1 = length(t+dt)
  l0 = L0 + z_0
  if ti < Nperiod*5:
    l1 = L1 + z_0 + v_0 * dt
  else:
    l1 = l_period[(ti+1) % Nperiod]
  dLdtt0 = dlengthdtt(t)
  dLdtt1 = dlengthdtt(t+dt)

  # Initial guess for pressure on structure
  if ti < Nperiod*5:
    if ti < 3:
      pGamma = pGamma_period[(ti-1) % Nperiod]
    else:
      pGamma = + 3 * pGamma_period[(ti-1) % Nperiod] \
               - 3 * pGamma_period[(ti-2) % Nperiod] \
               + 1 * pGamma_period[(ti-3) % Nperiod]
  else:
    pGamma = pGamma_period[ti % Nperiod]

  # Fluid-Structure loop
  for loop_k in xrange(2000):

    # Calculate structure response
    z_1, v_1 = calculate_structure(pGamma, dLdtt0, dLdtt1)

    # Stop fluid-structure iterations when converged
    if loop_k > 0:
      l1_res = abs(l1 - (L1 + z_1))
      #if not UseApprox:
      #  print "\t  %2d: (%7.3f)" % (loop_k, log10(l1_res))
      if l1_res < residual_tolerance:
        break
    l1 = L1 + z_1

    if UseApprox:
      # Calculate pressure on structure with approximate model
      pGamma_hat = CFD_approx.calculate_pressure(l0, l1)
      pGamma = pGamma_hat + pDelta[ti % Nperiod]
    else:
      # Calculate pressure on structure with CFD model
      pGamma = CFD_spacetime.calculate_pressure(l0, l1)

  # Update ti_periodic if solution is not yet periodic
  if abs(pGamma - pGamma_period[ti % Nperiod])/pGamma > periodicity_tolerance:
    ti_periodic = ti
  #print "t = %.4f \t " % t, "l = %.4f - %.4f \t pGamma = %.3f" % (l0, l1, pGamma/A)

  modelvariables.l_all[ti] = (l0 + l1) / 2
  l_period[(ti+1) % Nperiod] = l1
  modelvariables.pGamma_all[ti] = pGamma
  pGamma_period[ti % Nperiod] = pGamma

  # Advance timestep
  modelvariables.ti = ti + 1
  modelvariables.t = t + dt
  z_0, v_0 = z_1, v_1


def do_timesteps_until_periodic(l_period, pGamma_period, pDelta, UseApprox=False, PrescribeLength=False):
  import modelparameters
  print "DL =%6.3f : " % modelparameters.DL,
  while modelvariables.t < modelparameters.t_end-dt/2:
    if PrescribeLength:
      do_timestep_prescribed_l(l_period, pGamma_period)
    else:
      do_timestep(l_period, pGamma_period, pDelta, UseApprox)
    #if UseApprox and modelvariables.ti > 4*Nperiod and ti_periodic < modelvariables.ti - Nperiod:
    #  break
    if isnan(average(pGamma_period)):
      break
  pGamma_avg = average(pGamma_period) / A
  print "pGamma =%6.3f (%5.1f)\t(periods:%4.1f)" % \
    (pGamma_avg, log10(abs(pGamma_avg-1.6)/1.6), modelvariables.t*omega_L/(2*pi))
  return pGamma_avg
```

## modelproblem.py

```
from numpy import inf, isnan, logical_not, logical_or, average, sum as asum, dot, transpose, clip, log, ndarray
from numpy import zeros, ones, atleast_1d, atleast_2d, atleast_3d, ravel, vstack, hstack, concatenate, flatnonzero
from numpy.linalg import lstsq
from numpy.ma import MaskedArray
from scipy.optimize import fmin_powell
import string

import FSI_timestepping
import modelparameters
from modelparameters import A, p_S, l_S, Nperiod

target_pGamma_D = 1.6 * p_S

RelaxFixedZ = False
RelaxFixedDeltaP = False
RelaxFixedPhi = 0.8
RelaxAitken = False

UseArAdjust = 0
ArCoefficientType = 1
PDeltaAdjustmentType = 1
ArOrderP = 1

def reset_coupling():
  global convergence_history, phi_hist, pDelta_increment, pGamma_period_history
  convergence_history = []
  pDelta_increment = zeros([50, Nperiod])
  pGamma_period_history = []
  phi_hist = []
  global l_period, pGamma_period, pDelta
  l_period = zeros([Nperiod])
  pGamma_period = zeros([Nperiod])
  pDelta = zeros([Nperiod])
reset_coupling()


def do_timesteps_until_periodic(UseApprox=False):
  global l_period, pGamma_period, pDelta
  reset_coupling()
  FSI_timestepping.reset_time()
  return FSI_timestepping.do_timesteps_until_periodic(l_period, pGamma_period, pDelta, UseApprox=UseApprox)


def do_find_parameter_loose(n_max=50):
  global pDelta
  DL_D_approx = modelparameters.DL_D
  for n in range(n_max):
    if (UseArAdjust > 0 and n > 1 and ((n-2) % UseArAdjust) == 0) or (UseArAdjust < 0 and n == -UseArAdjust):
      def AR(phi, sequence):
        retval = phi[0] * sequence[-1]
        for m in range(1, min(len(phi), len(sequence))):
          retval = retval + phi[m] * sequence[-1-m]
        return retval
      print "determining phi...",

      # Determine coefficients used for pDelta adjustment
      if ArCoefficientType == 1:              # Use last averages
        P1, P2 = average(pDelta_increment[n-2,:]), average(pDelta_increment[n-1,:])
        phi = P2/P1
      elif ArCoefficientType == 2:            # Use last complete vectors
        Ps1, Ps2 = pDelta_increment[n-2,:], pDelta_increment[n-1,:]
        phi = dot(Ps1, Ps2) / dot(Ps1, Ps1)
      elif ArCoefficientType == 3:            # Use all averages
        Ps1, Ps2 = average(pDelta_increment[0:n-1,:], 1), average(pDelta_increment[1:n,:], 1)
        phi = dot(Ps1, Ps2) / dot(Ps1, Ps1)
      elif ArCoefficientType == 4:            # Use all complete vectors
        Ps1, Ps2 = ravel(pDelta_increment[0:n-1,:]), ravel(pDelta_increment[1:n,:])
        phi = dot(Ps1, Ps2) / dot(Ps1, Ps1)
      elif ArCoefficientType == 5:            # Use least squares AR (same as #4, but with possible higher order AR)
        def RMS_AR(phi):
          vector_AR = atleast_1d([(AR(phi, pDelta_increment[0:i,:]) - pDelta_increment[i,:]) for i in range(1, n)])
          return asum(vector_AR ** 2) + asum(phi ** 2) * 1e-14
        minres = fmin_powell(RMS_AR, zeros(ArOrderP), full_output=1)
        phi = atleast_1d(minres[0])
      else:
        phi = zeros([Nperiod, Nperiod+1])
        PP1 = hstack([pDelta_increment[0:n-1,:], transpose(atleast_2d(average(pDelta_increment[0:n-1,:], 1)))])
        PP2 = pDelta_increment[1:n,:]
      if ArCoefficientType == 6:                # VAR using constant diagonal (same as #4 with adjustment #3)
        PP1r = ravel(PP1[:,0:Nperiod])
        phi[xrange(Nperiod),xrange(Nperiod)] = dot(PP1r, ravel(PP2)) / dot(PP1r, PP1r)
      if ArCoefficientType == 7:                # VAR using diagonal
        for i in xrange(Nperiod):
          phi[i,i] = clip(dot(PP1[:,i], PP2[:,i]) / dot(PP1[:,i], PP1[:,i]), -0.95, 0.95)
      if ArCoefficientType >= 8 and ArCoefficientType < 9:      # VAR using average
        PP1_avg = PP1[:,Nperiod]
        for i in xrange(Nperiod):
          phi[i,Nperiod] = dot(PP1_avg, PP2[:,i]) / dot(PP1_avg, PP1_avg)
        PP1_all = PP1[:,:Nperiod]
        for m in xrange(int((ArCoefficientType-8)*10+0.5)):
          PP2_all = PP2 - transpose(dot(phi, vstack([transpose(PP1_all), atleast_2d(average(PP1_all, 1))])))
          phi_all = dot(transpose(PP2_all), PP1_all) / asum(PP1_all*PP1_all, 0)
          j_error_all = asum( (phi_all * atleast_3d(PP1_all) - atleast_3d(PP2_all))**2 , 0)
          min_errors_j = MaskedArray(j_error_all, logical_or(abs(phi_all)>1.0, phi[:,:-1]!=0.0)).argmin(axis=1)
          for i in xrange(Nperiod):
            phi[i,min_errors_j[i]] = phi_all[i,min_errors_j[i]]
      if ArCoefficientType == 9:                # subset VAR
        for i in xrange(Nperiod):
          phi_mask = zeros([Nperiod+1], dtype='bool')
          for m in xrange(7):
            sigma2_ref = asum((dot(PP1, transpose(phi[i,:])) - PP2[:,i])**2, 0) + 2e-20
            AIC_ref = log(sigma2_ref/(n-1)) + 2.0 / (n-1) * m
            sigma2 = ones([Nperiod+1]) * inf
            for j in flatnonzero(logical_not(phi_mask)):
              phi_mask[j] = True
              try:
                phi_masked = lstsq(PP1[:,phi_mask], PP2[:,i], rcond=0.1)[0]
                sigma2[j] = asum((dot(PP1[:,phi_mask], transpose(phi_masked)) - PP2[:,i])**2, 0) + 2e-20
              except:
                sigma2[j] = inf
              if j == Nperiod: sigma2[j] = sigma2[j] - 1e-20
              phi_mask[j] = False
            CannotImprove = False
            for j in sigma2.argsort():
              AIC_j = log(sigma2[j]/(n-1)) + 2.0 / (n-1) * (m+1)
              if AIC_j > AIC_ref and flatnonzero(phi[i,:]).size > 0:
                CannotImprove = True
                break
              phi_mask[j] = True
              phi_masked_new = lstsq(PP1[:,phi_mask], PP2[:,i], rcond=0.1)[0]

              # This is stricter than necessary, but using eigenvalues does not really work
              if all(abs(phi_masked_new[flatnonzero(phi_mask)<Nperiod]) < 1.0):
                phi[i,phi_mask] = phi_masked_new
                break
              phi_mask[j] = False
            if CannotImprove:
              break
          #print "row %2d: [%s]" % (i, string.join(['x' if b else '_' for b in phi_mask], ''))

      phi_hist.append(average(phi))
      print "done, phi:", ("%s, nonzeros: %d" % (phi.shape, flatnonzero(phi).size) if isinstance(phi, ndarray) else phi)

      # Adjustment of pDelta
      if PDeltaAdjustmentType == 1:         # Average additive
        pDelta = pDelta + phi / (1.0 - phi) * average(pDelta_increment[n-1,:])
      elif PDeltaAdjustmentType == 2:       # Average multiplicative
        pDelta = pDelta + phi / (1.0 - phi) * average(pDelta_increment[n-1,:]) * (pDelta / average(pDelta))
      elif PDeltaAdjustmentType == 3:       # Vector adjustments
        pDelta = pDelta + phi / (1.0 - phi) * pDelta_increment[n-1,:]
      elif PDeltaAdjustmentType == 5:       # Vector adjusments (same as #3, but with possible higher order AR)
        last_pDelta_increment = list(pDelta_increment[0:n,:])
        for i in xrange(100):
          last_pDelta_increment.append(AR(phi, last_pDelta_increment))
          pDelta = pDelta + pDelta_increment[-1]
      elif PDeltaAdjustmentType == 8:       # VAR
        last_pDelta_increment = pDelta_increment[n-1,:].copy()
        for i in xrange(100):
          last_pDelta_increment = dot(phi, concatenate([last_pDelta_increment, [average(last_pDelta_increment)]]))
          pDelta = pDelta + last_pDelta_increment

    global pGamma_period
    # Find solution of trimmed simple model
    DL_D_approx = do_find_parameter(DL_D_approx, UseApprox=True)
    pGamma_period_approx = pGamma_period.copy()

    if RelaxFixedZ:        # Relax on z
      global l_period, l_period_old
      if n > 0:
        phi = RelaxFixedPhi
        l_period = phi * l_period + (1.0-phi) * l_period_old
      l_period_old = l_period.copy()

    # Solve CFD with prescribed motion
    modelparameters.DL = DL_D_approx / l_S
```

```
    FSI_timestepping.reset_time()
    pGamma_D = FSI_timestepping.do_timesteps_until_periodic(l_period, pGamma_period, pDelta, PrescribeLength=True) * p_S

    # Update Delta p
    phi = 1.0
    if RelaxFixedDeltaP:    # Relax on Delta p with fixed phi
      phi = RelaxFixedPhi
    elif RelaxAitken:       # Use Aitken relaxation on Delta p
      if n > 0:
        R_n0 = pDelta_increment[n-1,:] / phi_hist[n-1]
        R_n1 = pGamma_period - pGamma_period_approx
        phi = - phi_hist[-1] * dot(R_n0, R_n1-R_n0) / dot(R_n1-R_n0, R_n1-R_n0)
      phi_hist.append(phi)
    pDelta = phi * (pGamma_period - pGamma_period_approx) + pDelta

    convergence_history.append((DL_D_approx, pGamma_D))
    if isnan(pGamma_D):
      return

    pDelta_increment[n,:] = phi * (pGamma_period - pGamma_period_approx)
    pGamma_period_history.append(vstack([pGamma_period, pGamma_period-pDelta]))
    print 'pDa: [%s]' % (','.join(['%7.4f' % (x/A) for x in average(pDelta_increment[0:n+1,:], 1)]))
    print 'phi_hist: [%s]' % (','.join(['%7.4f' % x for x in phi_hist]))

    if abs(pGamma_D - target_pGamma_D) / target_pGamma_D < 1e-12:
      break


DL_D_ref1 = 4.0
DL_D_ref2 = 6.0
def do_find_parameter(DL_D_initial, UseApprox=False, n_max=10):
  # Determine slope of DL-p line (sensitivity matrix)
  modelparameters.DL = DL_D_ref1 / l_S
  FSI_timestepping.reset_time()
  pGamma_ref1 = FSI_timestepping.do_timesteps_until_periodic(l_period, pGamma_period, pDelta, UseApprox=UseApprox)
  modelparameters.DL = DL_D_ref2 / l_S
  FSI_timestepping.reset_time()
  pGamma_ref2 = FSI_timestepping.do_timesteps_until_periodic(l_period, pGamma_period, pDelta, UseApprox=UseApprox)
  d_pGamma_d_DL_D = (pGamma_ref1 - pGamma_ref2) / (DL_D_ref1 - DL_D_ref2)

  # Trim iterations
  DL_D_loop = DL_D_initial
  for n in range(n_max):
    modelparameters.DL = DL_D_loop / l_S
    FSI_timestepping.reset_time()
    pGamma_D = FSI_timestepping.do_timesteps_until_periodic(l_period, pGamma_period, pDelta, UseApprox=UseApprox) * p_S
    convergence_history.append((DL_D_loop, pGamma_D))

    # Adjust trim
    DL_D_loop = DL_D_loop + (target_pGamma_D - pGamma_D) / p_S / d_pGamma_d_DL_D
    if DL_D_loop < 0.0 or isnan(DL_D_loop):
      print "*",
      DL_D_loop = 0.1

  return DL_D_loop
```