Lehrstuhl für
Strömungsmechanik
Friedrich-Alexander-
Universität
Erlangen-Nürnberg

MASTER THESIS

Dynamic calculation of under-relaxation factor in a partitioned, semi-implicit predictor-corrector coupling scheme for fluid-structure interaction solver

NARAYANAN ACHUTHAN

Erlangen, July 23, 2018

Examiner: [YOUR EXAMINER (probably Prof. Dr. Felix Freiling)]
Advisor: [YOUR ADVISOR]

Eidesstattliche Erklärung / Statutory Declaration

Hiermit versichere ich eidesstattlich, dass die vorliegende Arbeit von mir selbständig, ohne Hilfe Dritter und ausschließlich unter Verwendung der angegebenen Quellen angefertigt wurde. Alle Stellen, die wörtlich oder sinngemäß aus den Quellen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

I hereby declare formally that I have developed and written the enclosed thesis entirely by myself and have not used sources or means without declaration in the text. Any thoughts or quotations which were inferred from the sources are marked as such. This thesis was not submitted in the same or a substantially similar version to any other authority to achieve an academic grading.

Der Friedrich-Alexander-Universität, vertreten durch den Lehrstuhl für Informatik 1, wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Arbeit einschließlich etwaiger Schutz- und Urheberrechte eingeräumt.

Erlangen, July 23, 2018

[YOUR NAME]

Abstract

The presented numerical study focuses on dynamical calculation of the under-relaxation factor during each sub-iteration step of the *Fluid-Structure Interaction* solver using adaptive scehemes. The adaptive schemes presented in the study are *Aitken's* $\triangle^2$ *method* and *steepest descent method*. The mentioned schemes have been found to be efficient, yet easy to implement. The implemented schemes have been validated by a numerical simulation of flow around an elastically mounted circular cylinder at a Reynolds number of 200. (Cite Zhou)

The calculations were performed on a 2-D 0-type curvilinear orthogonal grids containing a total of 120x100 control volumes.The FSI simulations were performed using a *semi implicit predictor-corrector scheme* for fluid-structure coupling.The *semi implicit predictor-corrector scheme* is a strong coupling scheme between flow and structural solver, while also maintaining the explicit time marching schemes. The simulations were carried out for different reduced damping coefficients (Sg) and for a mass ratio (M^*) of 1. These cases were simulated with constant under-relaxation factor, and with dynamic under-relaxation factor using aitken's $\triangle^2$ method and steepest descent methods. The results were compared and validated with (Breuer and Muensch) and (zhou),the results were in good agreement with these established numerical data. Average time taken for sub-iterations within the time steps were calculated, *Aitken's* $\triangle^2$ *method* was observed to be more efficient in accelerating the convergence.

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Fluid-Structure Interaction (FSI) problems describe the coupled dynamics of fluid mechanics and structure mechanics. They are classical multi-physics problems and its application is very vast. Numerical simulation of such problems is a cumbersome process. There are mainly two approaches in solving the FSI problems *Monolithic approach* and *Partitioned approach*. In the current study, the focus is on the partitioned approach for solving FSI problems. Partitioned analysis techniques are more popular than the fully coupled Monolithic solvers, as they have computational superiority over the Monolithic solvers. Partitioned solvers allows for the use of suitable discretization methods, and optimized solvers for modeling of both fluid and structure. In the current investigation, adaptive schemes such as *Aitken's* Δ^2 *method* and *Steepest descent/gradient method* are implemented into the FSI solver to predict the *under-relaxation factor* dynamically. The current study also involves integration of an *Artificial Neural Network (ANN)* within the FSI solver for dynamic prediction of under-relaxation factors. The calculations are performed on an in-house *finite volume* based *FORTRAN* solver, *FASTEST-3D*. A partitioned, semi-implicit predictor-corrector coupling scheme method is used for solving FSI problems.

1.1 Need for dynamic relaxation methods

Partitioned schemes uses two separate solvers for solving fluid and structural equations. Strongly coupled methods require several sub-iterations for a particular time-step. And by their nature, iterative solvers require a convergence criteria which determines the termi-

nation of the iterations. In order to aid the convergence of the iterative solvers, relaxation factors are used. They can either be over-relaxation or under-relaxation parameters, in this study the focus is on under-relaxation factors $\omega \in (0, 1)$.

The most basic and easiest implementation is to use a *constant under-relaxation* factor for every iteration. This is not the most effective method, but it is robust and easy to implement. However, a poorly chosen under-relaxation factor can lead to numerical instabilities and ultimately slow down convergence. This leads to higher computational time and effort. There are no universal guidelines for selecting an appropriate constant under-relaxation factor, because they depend not only on the physical processes being approximated, but also on the details of the numerical formulation. It is often selected based on trial-and-error method for each problem.

In order to alleviate the problem, several adaptive methods can be found in literature for performing sub-iterations in an efficient and robust fashion. Mok and Wall [23], Wall et al. [36] proposed the most basic and highly efficient approach, the fixed-point iteration method with dynamic relaxation. Wall et al. [36] compared the effectiveness of three different methods, the *Aitken relaxation method*, *Tschebyscheff relaxation method* and *the method of steepest descent*. However, this work is only available in two hardly available conference proceedings and was never published in any journal.

Küttler and Wall [19] reproduced the work of Mok and Wall [23],and Wall et al. [36] where the treatment of both the *Aitken relaxation method* and *relaxation via the method of steepest descent* in the context of fixed-point iteration FSI solvers are studied. As per their study, the main advantage of fixed-point FSI solvers based on a Dirichlet–Neumann partitioning is the simple implementation with available field solvers. The calculation of a specific relaxation parameter in each iteration step influenced the convergence criteria. The relaxation methods were extended to FSI solvers based on Newton-Krylov methods. The Aitken relaxation method, proposed by Irons and Tuck [15] was found to be the most simple yet efficient method. Yigit et al. [38] investigated the efficiency of acceleration techniques for fluid structure interaction computations. The computations were carried out on a finite volume flow solver *FASTEST*, which is the solver used for the current study. Yigit et al. [38] implemented a multigrid procedure for moving meshes together with an adaptive under-relaxation strategy for accelerating the coupled computations. An Aitken relaxation method was integrated into the multigrid solver and the acceleration of the convergence were studied. The acceleration of the convergence improved with the integration of an adaptive under-relaxation technique.

Valdés et al. [34] implemented an algorithm to analyze the structures with large deformations, incompressible fluid flows and its fluid-structure interaction problems. This study emphasizes the use of *block-Gauss-Seidel method* with relaxation techniques. It has been observed that the strong coupling block Gauss-Seidel partition method with a relaxation method based on Aitken's method provided accelerated convergence for a variety of problems. Bogaers et al. [2] implemented a *Quasi-Newton method* with adaptive under-relaxation technique for implicit coupling of FSI solvers. It has been observed that for higher added mass ratios, the quasi-Newton methods with dynamic relaxation provides better results, when compared to fixed point iteration technique with dynamic relaxation.

1.2 Thesis objective

In the current study, the dynamic relaxation methods **Aitken's relaxation method** and **Steepest-Descent method** are implemented. The implementation is carried out on an in-house finite volume solver **FASTEST-3D**, a FORTRAN based solver. The coupling of the fluid and structure solver is done via a *semi-implicit predictor-corrector coupling*. Owing to the effectiveness and ease of implementation, these dynamic relaxation methods are selected and implemented. The FSI simulation of vortex induced vibration of a cylinder is validated and the effectiveness of the adaptive schemes is studied. The study replicates the work of Zhou et al. [39], and the results of the simulation are compared and validated. The setup replicates the spring mass damper system, as represented in the figure 1.1.

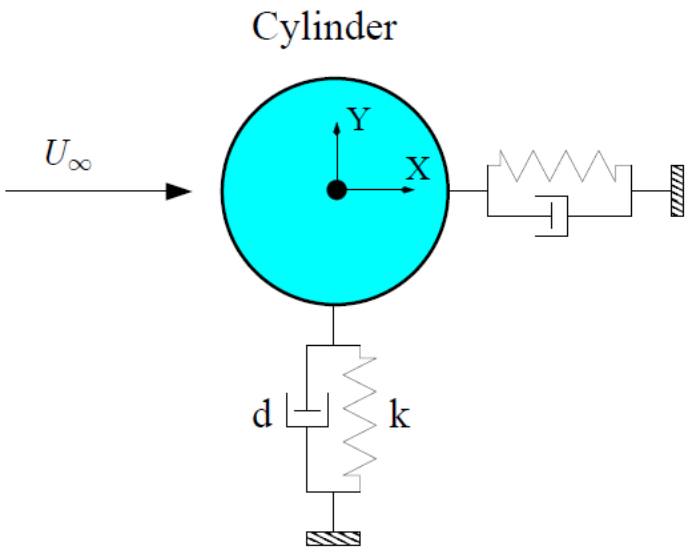


Figure 1.1: Representation of a spring-mass-damper system

The dynamic relaxation factors are then trained on a *Tensorflow MLP Regressor* neural network model. The trained model is then integrated externally via a shell script into the FASTEST-3D solver to predict the under-relaxation factors dynamically. The acceleration properties are studied and compared for these dynamic methods.

The thesis is structured as follows: An overview into the theoretical background related to the task is presented in the chapter 2. Chapter 3 discusses the numerical treatment of the methods used. The results from these methods are validated and presented in detail in chapter 4. Conclusive remarks are presented in chapter 5.

2 THEORETICAL BACKGROUND

2.1 Fluid-Structure Interaction

Fluid-Structure Interaction (FSI) studies the interaction between a structure (solid) and a fluid flow (liquid or gas) around it. It is a multi-physics problem which has large interest in diversified fields such as mechanical engineering (e.g. airfoils), civil engineering (e.g. towers) or medicine technique (e.g. artificial heart valves). Based on the response of structure and fluid fields, it is classified as one-way or two-way fluid-structure interaction problem. If the structural displacement/deformation does not influence the flow fluid or vice versa, then the FSI system is termed to be one-way fluid-structure interaction system. Conversely if the fluid flow and the displacement or deformation of the structure have significant influence on each other then the FSI system is termed as two-way fluid-structure interaction problem. Figure 2.1 represents a typical domain of a fluid-structure interaction problem, Ω refers to the common domain, Ω_f the fluid domain and Ω_s the structure domain. Figure 2.2a and 2.2b represents one-way and two-way FSI fluid-structure interaction problems respectively.

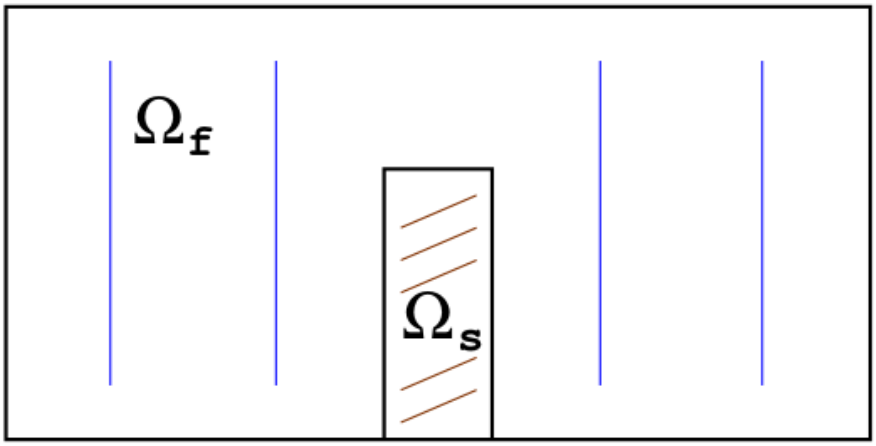
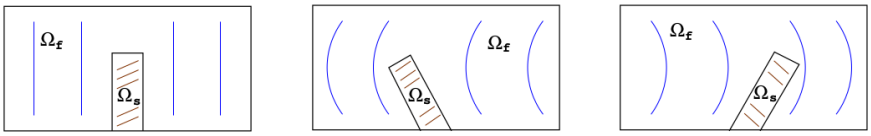
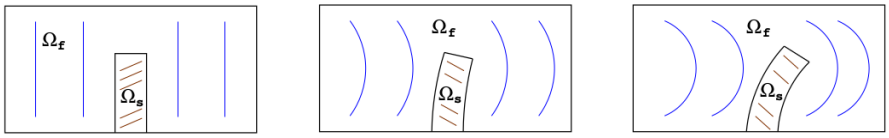


Figure 2.1: Representational figure of Fluid-Structure Interaction domain,taken from Richter [30]



(a) Structure domain influencing the fluid domain



(b) Both structure and fluid domains influencing each other

Figure 2.2: Representational figure of one-way 2.2a, and two-way 2.2b fluid-structure interaction domains,taken from Richter [30]

2.1.1 Numerical simulation strategies of FSI problems

Numerical simulation of FSI involves solving set of differential equations and corresponding boundary conditions for fluid and structural fields respectively. Suitable interface conditions needs to be defined so as the structural and fluid domains are well distinguished. There are different methodologies implemented to solve the FSI problem and are well documented in the literature. An overview of the numerical solution procedure for FSI problems are presented below.
The FSI problem is classified based on solution approaches and on the treatment of mesh

handling techniques as represented below. A brief overview of these methods are presented subsequently.

- Classification based on numerical solution approaches
  - Monolithic solver
  - Partitioned solver
    - Explicit coupling
    - Implicit coupling
- Classification based on meshing strategies
  - Conforming mesh
  - Non-conforming mesh

Classification based on numerical solution approaches

This classification is based on how the fluid and structural fields are getting solved. In *monolithic solvers* both the fluid and structural domains are expressed by single set of equations, whereas in *partitioned solver approach* structural fields and fluid fields are solved separately and additionally requires coupling of the distinct solvers. A brief introduction to these methods are represented below.

Monolithic solver approach

The equations governing the fluid flow and the displacement/deformation of the structure are represented by single set of equations which are then solved simultaneously by an unified algorithm, within a single solver framework Becker et al. [1], Hübner et al. [14], Michler et al. [22], Richter [30]. The central idea of monolithic solvers is to represent the interface by an homogeneous discretization, thus maintaining the conservation properties at the interface Michler et al. [22] Van Brummelen et al. [35]. Although monolithic approach gives strong coupling between the fields, it is commonly considered to be impractical for real world applications. It also demands enormous computational power for solving such a large system of equations since it has to incorporate the behavior of both fluid flow and solid structure. Figure 2.3 represents a schematic of the monolithic and partitioned solver approaches in solving FSI problems.

Partitioned solver approach

Another approach to fluid-structure interaction is to use two distinct solvers to model both fluid and solid domains. This technique allows the coupling of the fluid and solid solution by maintaining suitable coupling conditions. The interface conditions are used explicitly to communicate information between the fluid and structure solutions.

Some coupling algorithms were suggested by various studies (Farhat and Lesoinne [8], Felippa et al. [9], Piperno et al. [28]) which allows for reuse of existing codes that

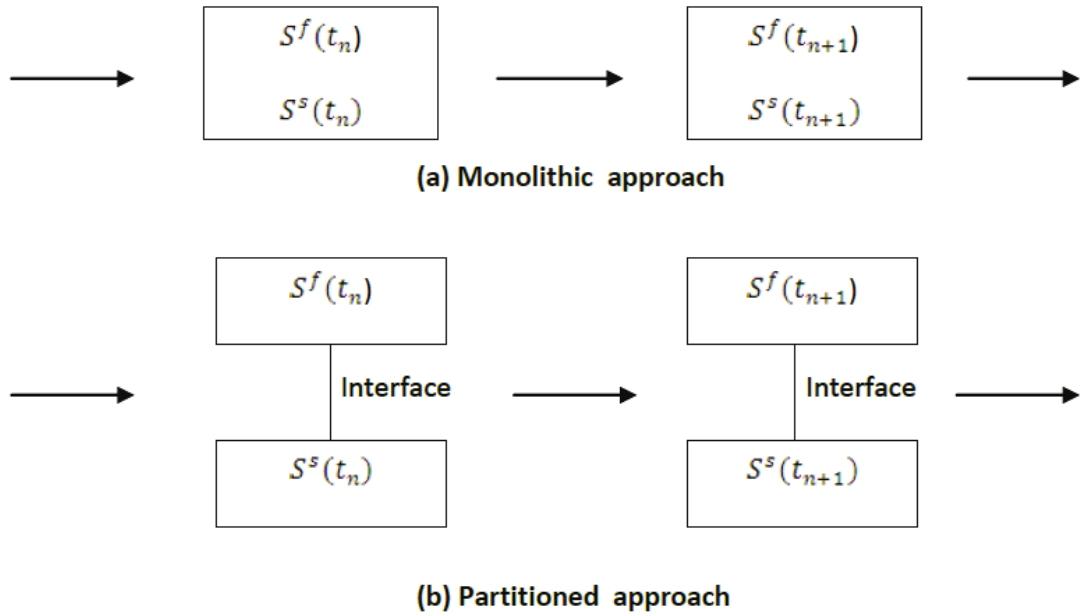


Figure 2.3: Schematic representation of Monolithic and Partitioned solution approaches. S^f represents the fluid solver and S^s represents the structural solver for two successive time steps current (t_n) and next (t_{n+1}). Figure taken from Hou et al. [13]

have been developed for each field. This approach is very robust and can be used for wide variety of applications. The major disadvantage of this approach is that, the interface location that divides the fluid and the structure domains is not known a-priori and usually changes in time. Thus, the partitioned approach requires tracking of the new interface location and its related quantities, which can be cumbersome and error-prone. The interface coupling conditions that are as stated below, have to be satisfied in order to have a stable solution.

- Kinematic coupling condition: The displacements, velocities and accelerations of the sub-zones have to be equal at the interface at any point in time.
  $\psi_\Gamma^{CFD}(t) = \psi_\Gamma^{CSD}(t), \dot{\psi}_\Gamma^{CFD}(t) = \dot{\psi}_\Gamma^{CSD}(t), \ddot{\psi}_\Gamma^{CFD}(t) = \ddot{\psi}_\Gamma^{CSD}(t)$
- Dynamic coupling condition: Conservation of the dynamic equilibrium of all forces at the interface needs to be satisfied. (Action and reaction forces must cancel out each other)
  $f_\Gamma^{CFD}(t) = -f_\Gamma^{CSD}(t)$

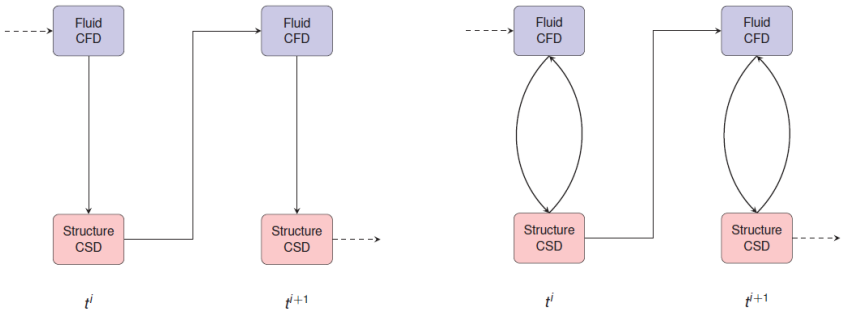
Depending on the influence of the structure movement on the fluid field the FSI problems are further subdivided into two following categories:

- Explicit/Weakly coupled: In an explicitly coupled algorithm, the equations of fluid mechanics, structural mechanics and the relative mesh movement are solved sequentially. Initially the governing equation of fluid is solved with the velocity

boundary condition derived from the structural displacement. The structural mechanics equations are solved next with the forces obtained from the fluid solver. Finally the grid is adapted based on the structural displacement. This coupling scheme is termed as weak coupling, as there is no sub iteration between the two sub zones.

- Implicit / Strongly coupled: In a strongly/implicitly coupled system, sub iterations between both the fluid and structural solvers are carried out at every time step till convergence is achieved. The convergence is determined by maintaining a small enough structural residuum. Detailed explanation on this, is presented in the subsequent section. This approach is implicit in nature and more robust, however requires additional computation time.

Figure 2.4 represents the schematic representation of explicit and implicitly coupled, partitioned approach to solving fluid structure interaction. In the presented study *semi-implicit predictor-corrector* coupling scheme is used.



(a) Explicit coupling **(b)** Implicit coupling

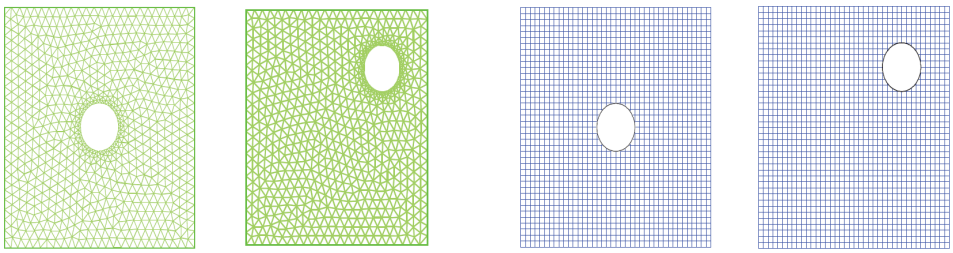
Figure 2.4: Representational sketch of explicit coupling 2.4a, and implicit coupling 2.4b approaches to solving partitioned, fluid-structure interaction problems,taken from Münsch [25]

2.1.2 Classification based on meshing strategies

The FSI problem is further classified based on the meshing strategies implemented to couple the fluid and structure domains. They are classified into two methods:*conforming mesh* and *non-conforming mesh*

- Conforming mesh method: In conforming mesh methods, the interface governing the fluid and structural domains is resolved such that the node connectivity is maintained between the domains. In this method, the interface conditions is considered as physical boundary conditions, which treat the interface location as part of the solution. Owing to the movement and/or deformation of the solid structure,remeshing (or mesh-update) is required.

- Non-conforming mesh method: The non-conforming mesh methods treat the interface location as constraints imposed on the model equations so that non-conforming meshes (node-connectivity need not be maintained between the domains) can be employed. As a result, the fluid and solid equations can be conveniently solved independently from each other with their respective grids, and re-meshing is not necessary. The distinction between these two types of meshes can be observed in figure 2.5, where a solid body (a sphere) is moving in a fluid domain.



(a) Conforming mesh method representation at time t_n and t_{n+1}

(b) Non-conforming mesh method representation at time t_n and t_{n+1}

Figure 2.5: Examples of Conforming mesh 2.5a and Non-conforming mesh 2.5b methods, taken from Hou et al. [13].

2.2 Governing equations of fluid flow and structural deformations

In Fluid-Structure Interaction problems, the solid moves through the fluid domain due to different forces or excitation. The computation domain changes with time and must be considered during the simulation. An approach which aims to solve this particular problem is Arbitrary Eulerian Lagrangian formulation, popularly referred to ALE formulation. The governing equation for conservation of mass and momentum are reformulated for a moving grid which are time dependent control volume and surface integrals. The governing equations for a Newtonian incompressible fluid in ALE formulation is given as follows.

$$\frac{\partial u_j}{\partial x_i} = 0 \tag{2.1}$$

$$\frac{\partial u_i}{\partial t} + (u_i - v_{g,i})\frac{\partial u_i}{\partial x_j} = \frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.2}$$

$$\frac{1}{J}\frac{\partial J}{\partial t} - \frac{\partial v_{g,i}}{\partial x_i} = 0 \tag{2.3}$$

The equations presented are *conservation of mass*,*conservation of momentum* and *Space*

Conservation Law (proposed by Demirdžić and Perić [5]) in ALE formulation in order to account for the moving grid representation. Here v_g is the grid cell velocity and J is the determinant of the metric tensor. Numerical treatment of these equations are presented in refer to the relevant section.

2.2.1 Coupling of fluid and structural solvers

The FSI problem in this study is solved by a *semi-implicit predictor corrector* coupling scheme, which is a strong coupling method between fluid and structural solver. Details about this scheme is explained in Breuer et al. [3], summary of this scheme is presented in brief as follows:

(a) At the start of the time step, the displacement of the structure is predicted based on values from the previous time step.

(b) A predictor corrector approach is implemented to solve for conservation of momentum and mass (velocity u_i and pressure p). This is explained in detail in section refer to relevant section

(c) The calculated forces are then tranferred to the Computational Structural Dynamics (CSD) solver. Generalized-α method is used to solve the conservation equations of the structural solver. This is explained in detail in section refer to relevant section.

(d) The FSI-subiterations are performed until desired structural and fluid convergence is achieved for a particular time step. Dynamic calculation of the under-relaxation factor is used to enhance the convergencerefer to relevant section. The corrector step takes the predicted velocity $\overset{\star}{u}_i$ as approximation to obtain the new corrected velocity u_i and pressure p.

(e) Mesh adaptation for each FSI-subiteration is performed, based on transfinite interpolation by Thompson et al. [33]. A comprehensive implementation can be found in Münsch and Breuer [24].

The predcitor-corrector algorithm for a strongly coupled, partitioned FSI solver is represented in the flowchart 2.6

Convergence of the FSI sub-iterations

As explained in flowchart 2.6 for an implicitly coupled, partitioned FSI solver, the flow field is determined in the actual flow geometry. From this, the friction and pressure forces on the interacting walls are computed. These are boundary conditions to the structural solver. The structural solver computes the deformations, with which the fluid mesh is then modified. Afterwards the flow solver is started again.

The FSI sub-iteration loop is repeated until a convergence criteria ϵ is satisfied, which is determined by the variation of the mean displacements 2.4.

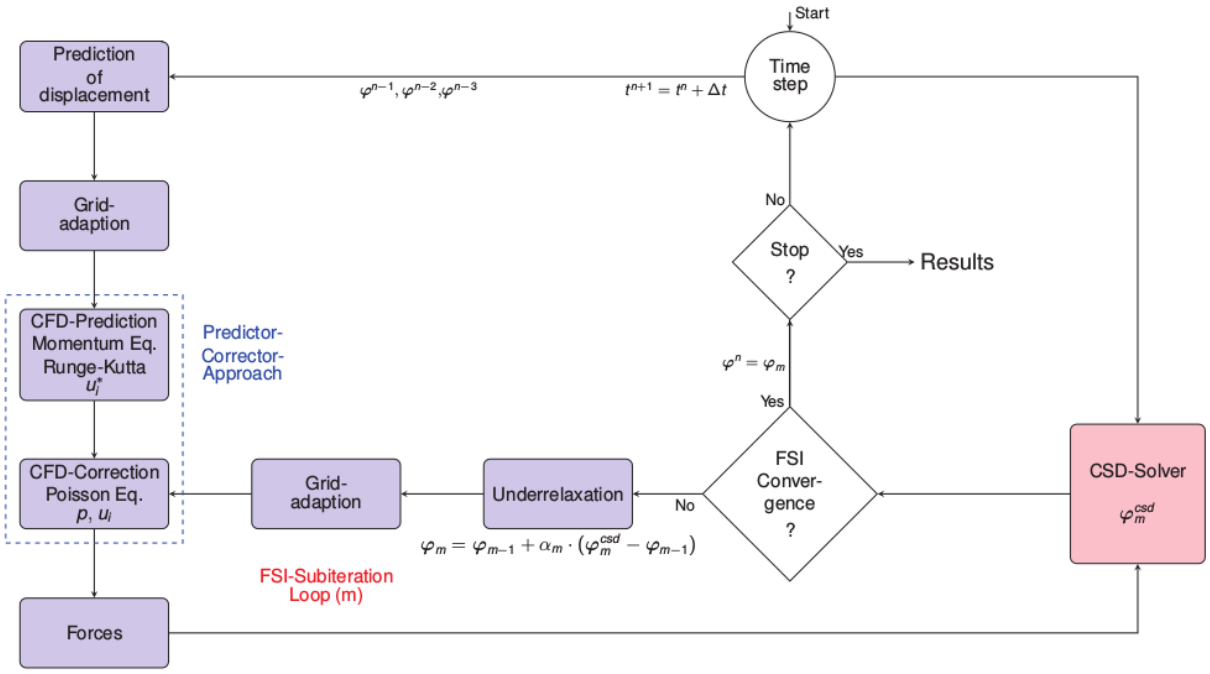


Figure 2.6: Flowchart representing the predictor-corrector coupling scheme to solve a strongly coupled, partitioned FSI solver Münsch and Breuer [24]

$$\frac{1}{N}\sum_{n=1}^{k}\frac{\left\|\mathbf{d}_i^{n+1}-\mathbf{d}_{i+1}^{n+1}\right\|^2}{\left\|\mathbf{d}_{i+1}^{n+1}-\mathbf{d}_{i+1}^{n}\right\|^2}<\epsilon \tag{2.4}$$

In order to accelerate the rate of convergence, a relaxation factor is added to the displacement and force values. The resulting displacement is computed from the current sub-iteration displacement value and displacement value from the last sub-iteration. Under-relaxation of the displacements are done as given by the equation 2.5, where ω represents the under-relaxation factor.

$$\mathbf{d}_{i+1}^{n+1} = \mathbf{d}_i^{n+1} + \omega_\mathbf{i}\left(\tilde{\mathbf{d}}_{i+1}^{n+1} - \mathbf{d}_i^{n+1}\right) \tag{2.5}$$

Relaxation of the interface displacements is synonymous to the line search step of the non-linear solver. There are different methods to calculate the under-relaxation parameter. The most simplest method is by using a *constant under-relaxation parameter* for all time-steps. There are various methods available now to determine the under-relaxation parameter dynamically, however the presented study focuses on *Aitken relaxation method* and relaxation via the method of *steepest descent*. An introduction to these methods is presented below.

(i) Constant under-relaxation method: The most simplest and ineffective method is to choose a constant ω for all time-steps. The relaxation parameter has to be small enough to keep the iteration from diverging, but as large as possible in order to use as much of the new solution as possible and to avoid unnecessary FSI iterations. The

optimal ω value is problem specific and not known a priori. Furthermore even the optimal fixed value could lead to more iterations than a suitable dynamic relaxation parameter.

(ii) Aitken relaxation method: It is an effective and cheap method to calculate the under-relaxation parameter dynamically. The central idea of Aitken's Δ^2 method is to use values from two previous iterations to improve the current solution, proposed by Irons and Tuck [15]. The aitken factor μ 2.6 is calculated from the displacement values, and the under-relaxation factor ω is calculated as given in the equation 2.9 Mok and Wall [23]. This method has been implemented and validated with many FSI applications Irons and Tuck [15], Küttler and Wall [19] .(Refer some more papers which uses aitken) The relaxation parameter calculation is valid in every FSI sub-iteration step. This method requires values from two previous sub-iterations, thus the relaxation parameter can be calculated after the first FSI sub-iteration. In the presented study the aitken factor from last time-step is used as an initial value for the current time-step $\left(\mu^n_{i_{max}} = \mu^{n+1}_1\right)$ as proposed by Irons and Tuck [15].

$$\mu_i^{n+1} = \mu_{i-1}^{n+1} + \left(\mu_{i-1}^{n+1} - 1\right) \frac{\left(\Delta\mathbf{d}_i^{n+1} - \Delta\mathbf{d}_{i+1}^{n+1}\right)^T \Delta\mathbf{d}_{i+1}^{n+1}}{\left(\Delta\mathbf{d}_i^{n+1} - \Delta\mathbf{d}_{i+1}^{n+1}\right)^T \left(\Delta\mathbf{d}_i^{n+1} - \Delta\mathbf{d}_{i+1}^{n+1}\right)} \tag{2.6}$$

$$\mathbf{d}_i^{n+1} = \mathbf{d}_{i-1}^{n+1} - \mathbf{d}_i^{n+1} \tag{2.7}$$

$$\mathbf{d}_{i+1}^{n+1} = \mathbf{d}_i^{n+1} - \tilde{\mathbf{d}}_{i+1}^{n+1} \tag{2.8}$$

$$\omega_i = 1 - \mu_i^{n+1} \tag{2.9}$$

(iii) Steepest-Descent method: The best relaxation parameter possible is the one that finds the optimal step length in r_{i+1}^{n+1} direction. In order to obtain that, the existence of a merit function φ is assumed, that is minimal at the solution d_{i+1}^{n+1} and sufficiently smooth, such that the relaxation parameter ω_i is given by equation 2.10. By Taylor's series expansion and assumption of connection between the merit function and interface residual, the final expression for the relaxation parameter is given by the equation 2.11. The steps in obtaining this expression is explained clearly in Küttler and Wall [19].

$$\omega_\mathbf{i} = arg \min_{\omega_i} \varphi(d_i^{n+1} + \omega_i r_{i+1}^{n+1}) \tag{2.10}$$

$$\omega_\mathbf{i} = -\frac{\left(\mathbf{r}_{i+1}^{n+1}\right)^T \left(\mathbf{r}_{i+1}^{n+1}\right)}{\left(\mathbf{r}_{i+1}^{n+1}\right)^T \mathbf{J} \left(\mathbf{r}_{i+1}^{n+1}\right)} \tag{2.11}$$

In the equation 2.11,**J** represents the Jacobian matrix of the interface residual $\left(\mathbf{J} = \varphi^{"}(\mathbf{d}_i^{n+1})\right)$. This interface Jacobian is not readily available, there are methods to obtain the matrix vector product which can be performed with black-box solvers Küttler and Wall [19]. In the presented study, the matrix vector product is obtained by means of Finite-Difference method as expressed in equation 2.12.

$$\mathbf{Jr}_{i+1}^{n+1} = \frac{\mathbf{fsi_{sub}}\left(\mathbf{d}_i^{n+1} + \delta\mathbf{r}_{i+1}^{n+1}\right) - \mathbf{d}_i^{n+1} - \delta\mathbf{r}_{i+1}^{n+1} - \mathbf{r}_{i+1}^{n+1}}{\delta} \tag{2.12}$$

$$\delta = \lambda\left(\lambda + \frac{\left|\mathbf{d}_i^{n+1}\right|}{\left|\mathbf{r}_{i+1}^{n+1}\right|}\right) \tag{2.13}$$

A small λ is assumed in the study, this method requires one more fsi-sub iteration cycle in order to evaluate the interface Jacobian. The computational costs for this method is slightly higher.

2.3 Flow separation and vortex shedding

The presence of the fluid viscosity slows down the fluid particles very close to the solid surface and forms a thin slow-moving fluid layer called a boundary layer. The flow velocity is zero at the solid surface to satisfy the no-slip boundary condition. Inside the boundary layer, flow momentum is quite low since it experiences a strong viscous flow resistance. Therefore, the boundary layer flow is sensitive to the external pressure gradient (as the form of a pressure force acting upon fluid particles). If the pressure decreases in the direction of the flow, the pressure gradient is said to be favorable. In this case, the pressure force can assist the fluid movement and there is no flow retardation. However, if the pressure is increasing in the direction of the flow, an adverse pressure gradient condition exists. In addition to the presence of a strong viscous force, the fluid particles now have to move against the increasing pressure force. Therefore, the fluid particles could be stopped or reversed, causing the neighboring particles to move away from the surface. This phenomenon is called the boundary layer separation or flow separation.
In fig.2.7 there exists a pressure gradient $\frac{\partial p}{\partial x} < 0$ to accelerate the fluid along the surface x from D to E and a gradient $\frac{\partial p}{\partial x} > 0$ to decelerate the fluid between E and F. Potential energy (PE) in the pressure field is converted to kinetic energy between D and E, then fully back to PE between E and F, because there is no source of dissipation in an inviscid fluid.

In the inviscid region, the fluid momentum is sufficient to overcome the adverse pressure gradient barrier and reach point F with the same kinetic energy as at point D. However, inside the boundary layer (near the surface of the cylinder) energy is dissipated by the viscous drag. As the momentum is smaller near the wall than in the regions closer to the free stream, a fluid particle on the boundary layer can not overcome the pressure barrier and comes to a standstill and eventually gets pushed backwards into motion by the pressure distribution of the outer flow. This continuous retardation of the flow brings the wall shear stress τ_w to zero at the point S on the wall as shown in figure 2.8 . The shear stress starts to become negative after the point S, which is the separation point. The boundary layer separation point is given by the mathematical relationship in equation 2.14.

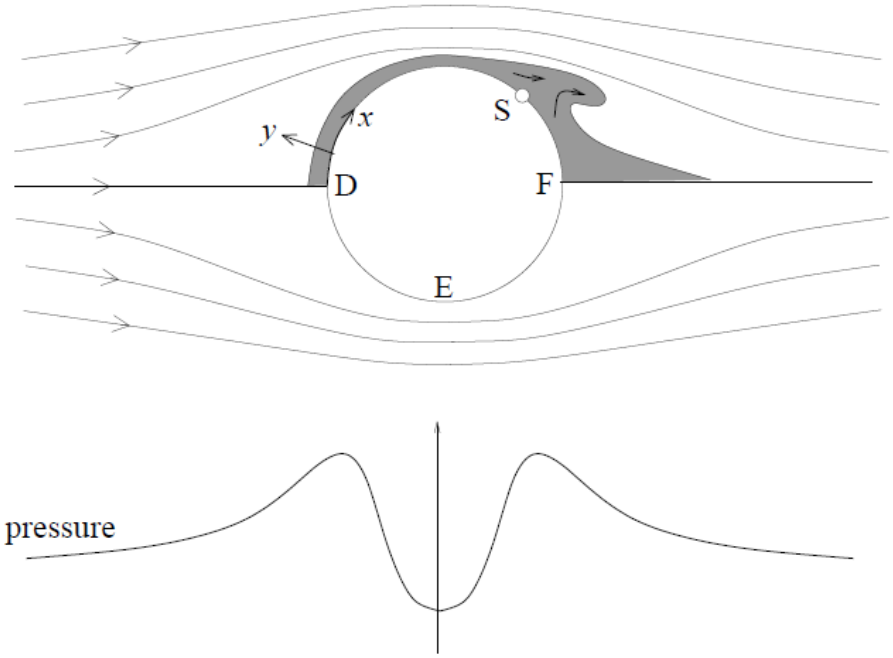


Figure 2.7: Pressure field and separation point for flow past a cylinder.

$$\tau_w = \mu \left(\frac{\partial u}{\partial y} \right) \tag{2.14}$$

Due to the existence of the high shear layer near the cylinder surface, the breaking mechanism is caused by Kelvin-Helmholtz instability. This instability leads to a rolling up and to a separation from the surface of the cylinder that result in the formation of the vortex street (Lin and Pauley [21], Nishioka and Sato [26], Posdziech and Grundmann [29]). This breaking mechanism is the basic physics concept of vortex formation and the near wake; that can be an origin of technique for triggering vortex. Vortex shedding occurs alternatively from top and bottom section of the cylinder. Vortex formation and separation for different Reynolds number (Re) is illustrated in figure 2.9.

2.4 Vortex Induced Vibrations

The periodic vortex shedding from a bluff body in a flow can cause the structure to vibrate. The vibrations alter the vortex pattern and change the spacing between vortices in the wake. These types of vibrations is called as *Vortex Induced Vibrations*. The case of an elastically-mounted circular cylinder, is one of the most conceptually basic cases of this phenomenon. The cylinder acts as a rigid body, translating through the fluid, while its elastic supports deform to accommodate the motion. The studies of Feng [10], Griffin [11], covered a wide range of *Reynolds number* and show that the vortex-induced vibration is a self-limiting process. The cross-flow vibration amplitude has a strong relationship with the phase difference between the lift force and the cylinder motion. According to the

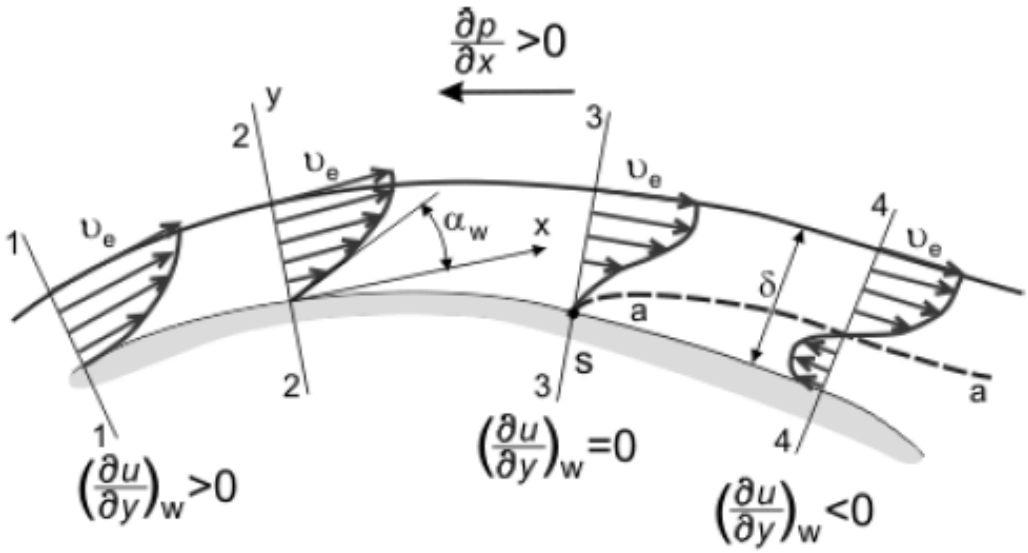
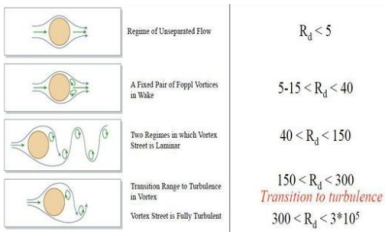
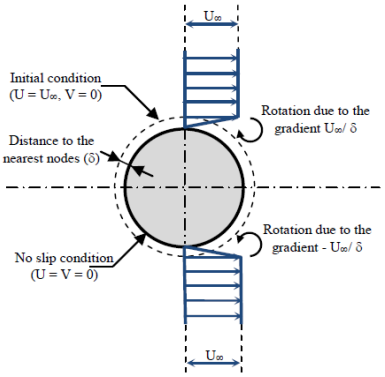


Figure 2.8: point of separation and Wall Shear stress,ref Schlichting et al. [32]



(a) Vortex shedding influenced by Reynolds number



(b) Vortex formation

Figure 2.9: Vortex formation and shedding phenomenon, ref Laroussi and Djebbi [20]

study of Williamson [37], the upper limit of Reynolds number, where the vortex shedding from a circular cylinder is still two-dimensional and the wake is laminar, can reach 200 under carefully controlled experimental conditions.
In the case of a cylinder treated as an elastically mounted spring-damper-mass system, the motion of the cylinder in perpendicular axis, is described by the equation 2.15.

$$m\ddot{y} + d\dot{y} + ky = F_y \tag{2.15}$$

In the above equation, y,$\dot{y}$,$\ddot{y}$ represents the displacement, velocity and acceleration components in the direction of lift force respectively. m,d and k represents the structural mass, damping co-efficient and spring stiffness constant of the system. These parameters are defined by the reduced damping parameter $S_g = 8\pi^2 S_t^2 M^\star$, which represents the vibration amplitude and was proposed by Griffin [12]. The mass ratio $M^\star$ is a dimensionless pa-

rameter which is the ratio of structural mass to mass of the fluid. $M^{\star} = \frac{m_s}{\rho \mathbf{D^2 h}}$ where m_s represents the structural mass of the cylinder. The *Strouhal number* S_t is a dimensionless number which normalizes the Vortex shedding frequency f_s and is given by, $S_t = \frac{f_s D}{U_\infty}$. From the studies of Zhou et al. [39], the strouhal number of a fixed cylinder at Re=200 is found to be 0.195. The same value has been considered for this present study. The calculations of the relevant parameters is presented in the subsequent section.

2.4.1 Calculation of parameters influencing the vortex induced vibrations

(a) Structural Mass m_s: The structural mass m_s is given by the general mass formula:

$$m_s = \rho \frac{1}{4} \pi D^2 L \tag{2.16}$$

where ρ, L and D represents the density, length and diameter of the cylinder. In the present study,$\rho = 1kg/m^3$, L=D=1m, hence $m_s = \frac{\pi}{4} kg$.

(b) Spring Stiffness constant k: The stiffness constant of the system is defined by the equation:

$$k = m_{eff} \omega_n^2 \tag{2.17}$$

$$\omega_n = 2\pi f_n \tag{2.18}$$

$$k = m_{eff} \left(2\pi f_s\right)^2 \tag{2.19}$$

where ω_n represents the natural angular frequency of the system and m_{eff} is the effective mass which is the sum of structural mass and fluid mass. The frequency ratio $\left(\frac{f_n}{f_s}\right)$ of 1 is considered in this present study.

(c) Damping co-efficient d: The damping co-efficient of the spring-mass-damper system is calculated from the reduced damping parameter S_g, which is defined by the equation 2.20.

$$d = \frac{S_g}{8\pi^2 {S_t}^2 M^*} \tag{2.20}$$

The response of the system is studied for a Mass ratio M^* of 1 and different reduced damping parameter S_g. Table 2.1 represents the parameters calculated for different S_g, replicating the work of Zhou et al. [39].

	S_g	$k[N/m]$	$d[kg/s]$
$M^* = 1$	0.01	2.6802	0.0033
	0.1	2.6802	0.0333
	1	2.6802	0.3331

Table 2.1: Spring stiffness co-efficient and damping co-efficient calculated for $M^* = 1$ and different S_g values.

2.5 Artificial Neural Network

The simplest definition of a neural network, more properly referred to as an 'artificial' neural network (ANN), is provided by the inventor of one of the first neuro-computers, Dr. Robert Hecht-Nielsen.
"A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs",reference Caudill [4]
An ANN is built upon the principle of a biological *neuron*, which has *dendrites* to receive signals, a *cell body* to process them and an *axon* to communicate the signals to the other neurons. Similarly ANNs are made up of layers, it can have a number of *input layers*, a processing layer mostly referred to as *hidden layers* and an *output layer*. Figure 2.10 expresses the similarity between the biological neuron and an ANN.
Neural networks are typically organized in layers. Layers are made up of a number of

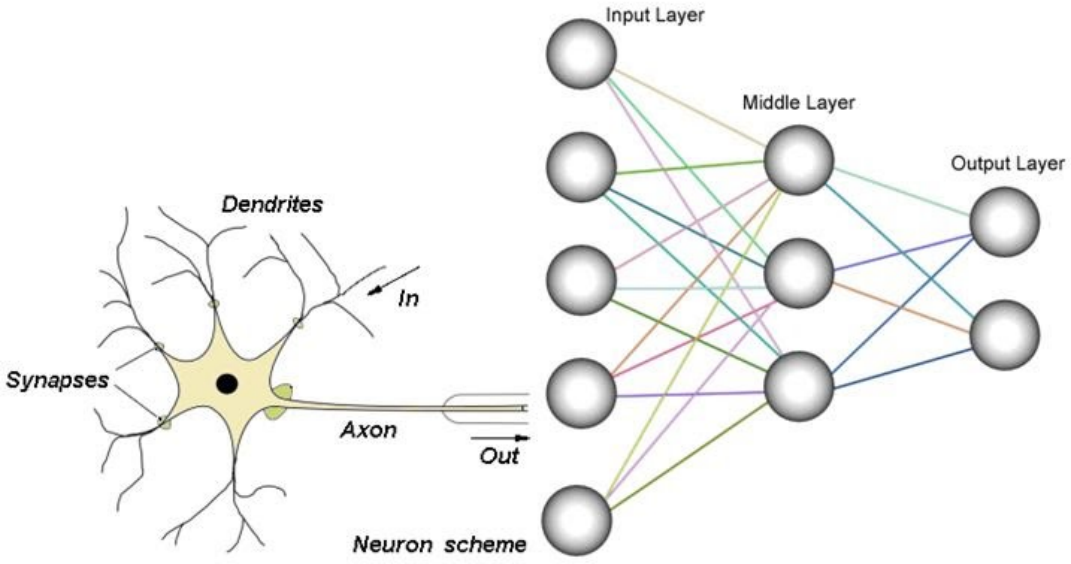


Figure 2.10: Representation of a biological Neuron and an Artificial Neural Network

interconnected neurons which contain an *activation function*. Patterns are presented to the network via the *input layer*, which communicates to one or more *hidden layers* where the actual processing is done by a system of *weighted connections*. The hidden layers then

link the result to an *output layer*. Each neuron element has a numeric threshold value. The neuron compares the effective weight to this threshold value. If the effective weight exceeds the threshold, the unit produces an output value of 1. If it does not exceed the threshold it produces an output of 0. The process of adjusting the weights an threshold values in a neural net is called **training**. A neural net is trained with significant data-set to produce the desired results. Figure 2.11 represents an example of weights and biases and an activation function.

The Artificial Neural Networks are very versatile and can be used for numerous applica-

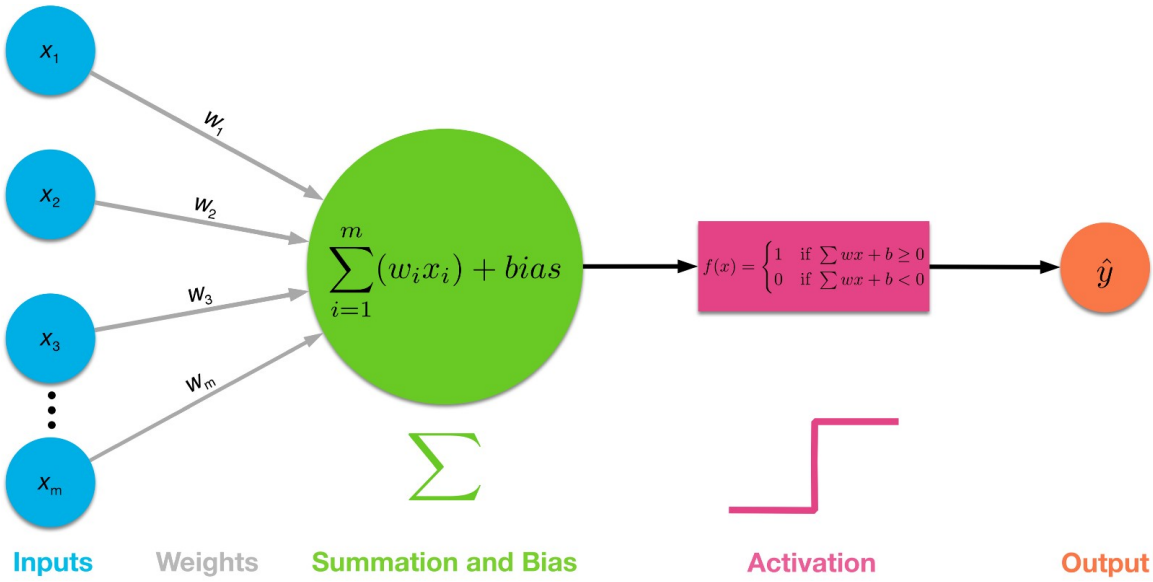


Figure 2.11: Representation of weights, biases and an activation function which constitutes the working of an Artificial Neural Network.

tions. Based on their application, they are classified into various categories. According to Jain et al. [16], most applications of neural networks fall into the following categories:

- Pattern Classification: The task of pattern classification is to assign an input (speech or image) to pre-described classes. Classic examples are character recognition, speech recognition etc.
- Clustering or Categorization: This particular type comes under *unsupervised learning*, where there are no training data with classified labels. The network explores the similarity between the patterns and places similar patterns into a cluster. Examples include data mining, data compression etc.
- Prediction or Forecasting: From a set of time dependent samples, the network is trained to predict or forecast a sample at a future time frame. *Stock market* and *weather forecasting* are examples of this type.

- Optimization: This is one of the most widely used technique in variety of fields such as mathematics, science, statistics, medicine and economics. The goal of an optimization algorithm is to minimize or maximize an objective function.

2.5.1 Learning of an Artificial Neural Network

The learning of an ANN can be attributed as the problem of updating network architecture and connection weights in order that the network can predict or perform a specified task. The network usually must learn the connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network. There are primarily three types of learning:

- Supervised learning: The network is provided with the output for every input pattern and the weights are adjusted iteratively in order to achieve the desired output.
- Reinforcement learning: Reinforcement learning is a variant of supervised learning in which the network is provided with only a critique on the correctness of network outputs, not the correct answers themselves.
- Unsupervised learning: Unsupervised learning, does not require an exact output or target associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations. Hybrid learning combines supervised and unsupervised learning. Part of the weights are usually determined through supervised learning, while the others are obtained through unsupervised learning.

In this study, the focus is on **Supervised learning**, where the deep neural network architecture is provided with inputs and associated targets.

Cost function

The neural network can be trained using batch gradient descent on a set of fixed training data $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$. The cost-function of a single training data can be expressed as in equation 2.21.

$$J(W, b; x, y) = \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 \tag{2.21}$$

The above equation represents a squared-error cost function $J(W, b; x, y)$, which is dependent on weights and biases of a single training data. For the set of m examples, the overall cost-function is defined to be:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \tag{2.22}$$

$$= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \tag{2.23}$$

The first term in the above equation is an average sum-of-squares error term. The second term in the equation is a regularization term (λ) also called as *weight decay term*, which tends to decrease the magnitude of weights and helps prevent over-fitting. The primarily goal of learning of an ANN is to minimize the cost function $J(W,b)$ as a function of weights and biases. The **Back-propagation algorithm** proposed by Rumelhart et al. [31] is used to train the neural network model.

Back-propagation algorithm

In order to train a Deep Neural Network (An ANN with one or more hidden layers), the weights W_{ij}^l connecting the nodes in layer i to layer j and corresponding biases b_i^l needs to be randomly initialized to some value distributed around zero. Then an optimization algorithm such as *steepest-descent method* or *Marquardt levenberg algorithm* is applied. The *back-propagation algorithm* can be applied only on *Feed-Forward* or *Multi Layer Perceptrons network*, and are ideal for Pattern Recognition and Regression models. The weights and biases of the network is updated backwards after an initial "forward pass". The algorithm is as described below:

- For a given training example (x,y), a *forward pass* to compute the activation throughout the network, including the output value of the hypothesis $h_{W,b}(x)$ is carried out. The activation for hidden layers $(L2, L3, \ldots, L_{nl-1}$ and the output layer L_{nl} is done during the forward pass. Here, nl represents the output layer.

- For each node i in layer l, an error term δ_i^l that quantifies the impact of a particular node on the errors in the output is defined. For an output node, the difference between the network's activation and the target value can be used to define δ_i^{nl}.

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}}\ \frac{1}{2}\left\|y - h_{W,b}(x)\right\|^2 = -(y_i - a_i^{(n_l)})\cdot f'(z_i^{(n_l)}) \tag{2.24}$$

- For the hidden layers, the computation of error term δ_i^l is based on a weighted average of the error terms of the nodes for a corresponding input node $a_i^{(nl)}$. For the hidden layers $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$ and for each node i in layer l, set:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)}\right) f'(z_i^{(l)}) \tag{2.25}$$

- After computing the error terms of all corresponding layers, the partial derivatives of the cost function is given by:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)} \tag{2.26}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}. \tag{2.27}$$

After the back-propagation algorithm, based on the optimizer the complete algorithm of update rule differs. The algorithm mentioned below is considering *Stochastic Gradient Method* as an optimizer. One iteration of the batch gradient descent is as follows:

- Considering $\Delta W^{(l)}$ and $\Delta b^{(l)}$ in the same dimensions of weights and biases, set: $\Delta W^{(l)} := 0$ and $\Delta b^{(l)} := 0$ for all layers l.
- For $i = 1$ to m,
  a) Back-propagation is done to obtain $\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y)$ and $\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y)$.
  b) Set $\Delta W^{(l)} := \Delta W^{(l)} + \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y)$
  c) Set $\Delta b^{(l)} := \Delta b^{(l)} + \frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y)$
- The weights and biases are updated:

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right] \tag{2.28}$$

$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right] \tag{2.29}$$

  where α is the *learning rate* of the network.

2.5.2 Factors influencing the learning of a network

There are certain factors which influence the learning of a neural network.

(i) Choice of optimizer

(ii) Activation function

(iii) Learning rate

(iv) Number of hidden layers

Choice of Optimizer

There are many optimizer methods available to minimize the *cost-function* such as *Stochastic Gradient Descent*, *Momentum*, *Adagrad, Adadelta, RMSprop,Adam* and Newton's optimization method *L-BFGS* etc. The choice of optimizer method is purely application oriented, and in the current study the optimizer selected is **adam**.

Adam optimizer This method is an update to the *stochastic gradient descent* approach which uses single learning rate for all weight updates, whereas in Adam the learning rate is adaptive. Adam is short for *"adaptive moment estimation* and was proposed by Kingma and Ba [17]. This method combines the advantages of two methods *AdaGrad* and *RMSProp*. The simplified form of the algorithm proposed by Kingma and Ba [17] is as follows:

$$m = \beta_1 m + (1 - \beta_1)\, dx \tag{2.30}$$

$$v = \beta_2 v + (1 - \beta_2)\left(dx^2\right) \tag{2.31}$$

$$x = x - \frac{\alpha m}{\sqrt{v} + eps} \tag{2.32}$$

where m and v are momentum vectors, and x is the position or point along the descent direction. As per the study of Kingma and Ba [17], the recommended values for exponential decay rates β_1 and $beta_2$ are 0.9 and 0.999 respectively. The effectiveness of this method can be observed from figure 2.13 taken from their study.

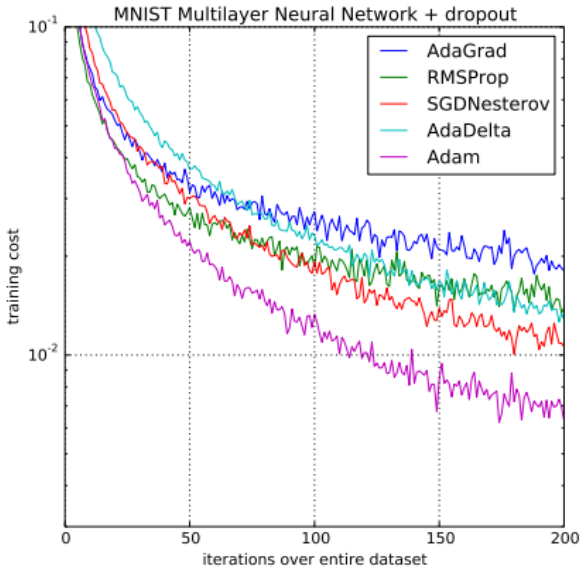


Figure 2.12: Performance of various optimizer on MNIST images data-set, taken from Kingma and Ba [17]

Activation Function

The activation function determines the output value of each neuron, it determines whether a particular neuron is activated or not. Duch and Jankowski [6] had proposed over 640 different activation functions. There are some common activation functions available such as *Perceptron activation*, *sigmoid function*, *Rectified Linear Units (ReLU)* and *hyperbolic tan function*. The purpose of an activation function in a deep neural network is to ensure that the representation in the input space is mapped to a different space in the output. A similarity function between the inputs and weights are performed by the neural network. In the current study **ReLU** activation function has been selected.

Rectified Linear Unit (ReLU) activation function: This is the most widely used activation function, across various applications. The function $\sigma(x) = max(0, x)$ returns only positive values for any given input x. This function does not require any normalization as the values returned are either 0 or positive. Figure 2.13 represents the functional

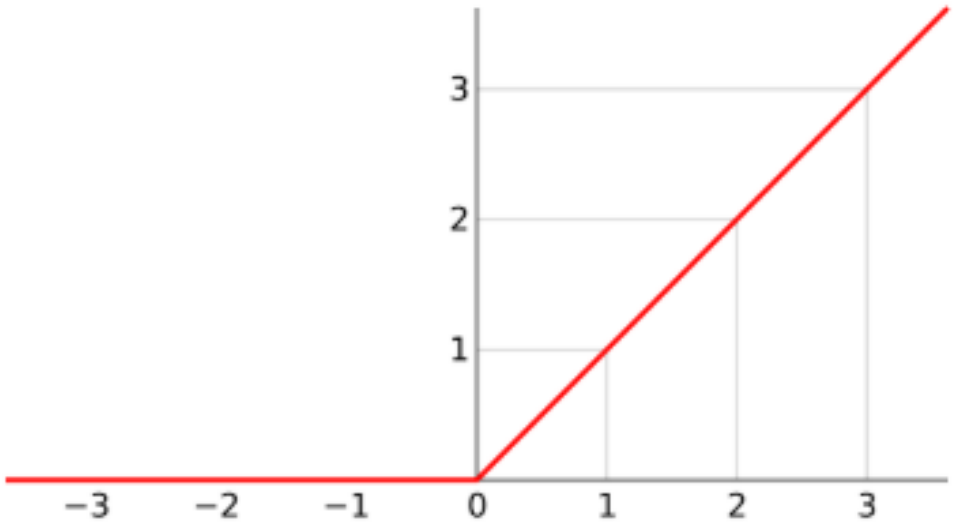


Figure 2.13: Representation of ReLU function output values for different input values

behavior of a ReLU activation function.

Learning rate

The learning rate has a direct impact on the loss function per epoch and ultimately the overall loss function of the training session. Low learning rates minimizes the loss linearly and it takes longer for the loss to decay to an acceptable value. High learning rates will decay the loss faster, however the overall loss value might be higher than the acceptable limit. Figure 2.14 depicts the effect of learning rate on training loss.

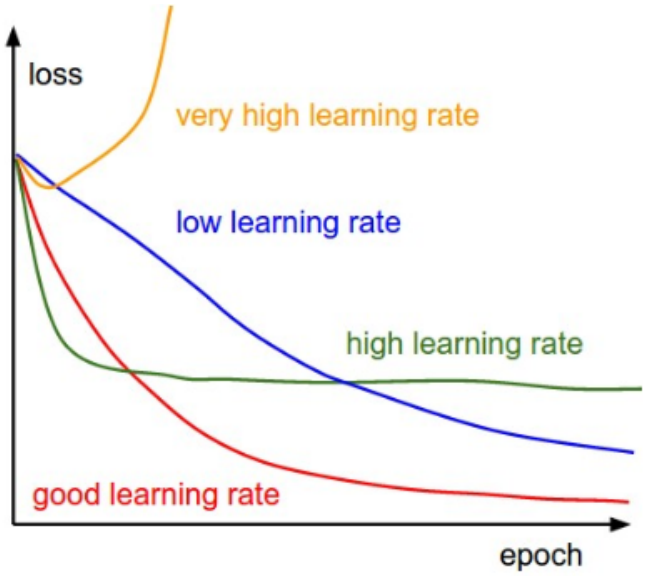


Figure 2.14: Effect of learning rates on training loss

Number of hidden layers

A deep neural network consists of one or more hidden layer units, in addition to the input and output layer. The hidden layers are required to fit the neural network model, $i.e$ to map any function from one finite space to another. However, the number of hidden layers is mostly application or problem specific and in this study 2 hidden layers are selected. Further number of hidden layers hinders the effectiveness of Back-propagation algorithm as the numerical error from one layer to the next and so on.

The parameters used to build and train the Neural Network model is presented in the table 2.2.

Network type	Multi Layer Perceptron model
Optimizer method	Adam optimizer
Activation function	ReLU
Learning rate	0.001
Number of hidden layers	2
Number of training epochs	500

Table 2.2: Features and parameters used to train the Tensorflow Neural Network model

3 IMPLEMENTATION OF THE TASK

3.1 Numerical treatment of the governing equations

The numerical results provided in this thesis study has been calculated in **FASTEST-3D** short for, *Flow Analysis by Solving Transport Equation Simulating Turbulence*. The solver is developed by the LSTM branch of Friedrich Alexander University, Erlangen-Nürnberg. The FASTEST-3D solver is a finite volume solver based on co-located, block-structured meshes.The solver code is written in FORTRAN and is highly parallelizable for use in modern multi-core processors. The numerical treatment of incompressible Navier Stokes equations generated in Cartesian co-ordinates is based on the procedure proposed by Peric [27]. It consists of a fully conservative, second-order finite volume space discretization with a collocated arrangement of variables on structural, multi-block grids. The procedure for coordinate transformation, Finite volume formulation and discretization schemes for the numerical solution of the governing equations are discussed in the following sections. These sections are summarized from the Doctoral dissertation of Kumar [18] and Münsch [25].

3.1.1 Coordinate transform

The concept of coordinate transformation is to represent a complex physical domain in a much more simpler, orthogonal computational domain. The numerical treatment of a orthogonal computational domain has many advantages over non-orthogonal domains. As a consequence of coordinate transformation all the governing equations have to be rep-

resented in a general curvilinear coordinate system. The solver currently supports only orthogonal domains, this thesis study also consists of orthogonal computational domain. The coordinate transformation from Cartesian to a curvilinear orthogonal coordinate system is based on the procedure suggested by Durst [7].

Assuming the physical space in the 3-D Cartesian coordinate system is represented by (x, y, z) and its corresponding domain in Curvilinear coordinate system is represented by (ξ, η, ζ) respectively. The relationship between the two coordinate systems are given by:

$$\xi = \xi(x, y, z) \tag{3.1}$$
$$\eta = \eta(x, y, z) \tag{3.2}$$
$$\zeta = \zeta(x, y, z) \tag{3.3}$$

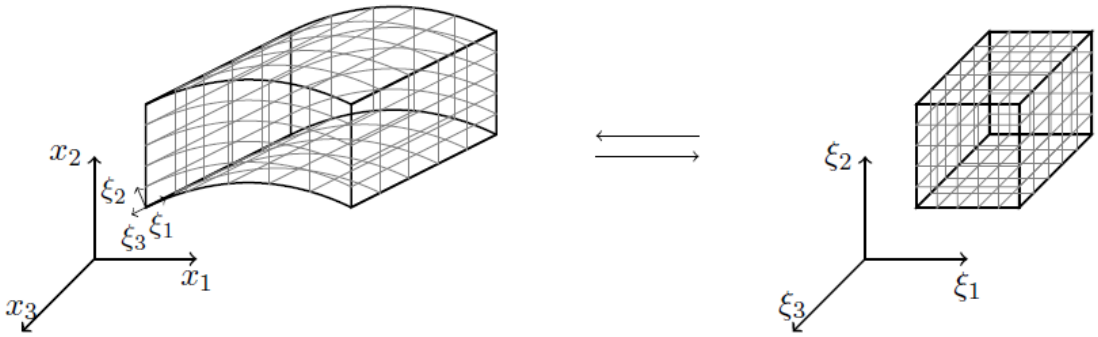


Figure 3.1: Representation of coordinate transformation from Cartesian physical space (left) to a curvilinear orthogonal computational space (right), figure taken from Münsch [25]

The differential operators of the curvilinear coordinate domain is given by:

$$d\xi = \frac{\partial \xi}{\partial x}dx + \frac{\partial \xi}{\partial y}dy + \frac{\partial \xi}{\partial z}dz \tag{3.4}$$
$$d\eta = \frac{\partial \eta}{\partial x}dx + \frac{\partial \eta}{\partial y}dy + \frac{\partial \eta}{\partial z}dz \tag{3.5}$$
$$d\zeta = \frac{\partial \zeta}{\partial x}dx + \frac{\partial \zeta}{\partial y}dy + \frac{\partial \zeta}{\partial z}dz \tag{3.6}$$

Similarly the reverse transformation from curvilinear space to the Cartesian space is denoted as follows.

$$x = x(\xi, \eta, \zeta) \tag{3.7}$$
$$y = y(\xi, \eta, \zeta) \tag{3.8}$$
$$z = z(\xi, \eta, \zeta) \tag{3.9}$$

The total differential operators for reverse transformation is given by the following

terms.

$$dx = \frac{\partial x}{\partial \xi} d\xi + \frac{\partial x}{\partial \eta} d\eta + \frac{\partial x}{\partial \zeta} d\zeta \tag{3.10}$$

$$dy = \frac{\partial y}{\partial \xi} d\xi + \frac{\partial y}{\partial \eta} d\eta + \frac{\partial y}{\partial \zeta} d\zeta \tag{3.11}$$

$$dz = \frac{\partial z}{\partial \xi} d\xi + \frac{\partial z}{\partial \eta} d\eta + \frac{\partial z}{\partial \zeta} d\zeta \tag{3.12}$$

Both the transformations can be represented in a matrix notation as given by the following set of equations.

$$\begin{pmatrix} d\xi \\ d\eta \\ d\zeta \end{pmatrix} = \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{pmatrix} \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} \tag{3.13}$$

$$\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{pmatrix} \begin{pmatrix} d\xi \\ d\eta \\ d\zeta \end{pmatrix} \tag{3.14}$$

The matrices in the equations 3.13 and 3.14 are termed as transformation or *Jacobian* matrices. The matrices are referenced as a_{ij} and b_{ij} respectively for easier notation. In general applications the computational grid is generated in the physical space. The relations in mapping to the curvilinear space as given by the set of equations 3.3 are unknown and consequently the transformation matrix a_{ij} as represented in the equation 3.13 is also unknown.

The transformation matrix a_{ij} can be calculated from the following relationship:

$$\begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{pmatrix}^{-1} \tag{3.15}$$

This can also be represented using referenced notation as

$$(a_{ij}) = (b_{ij})^{-1} \tag{3.16}$$

The inverse of the transformation matrix b_{ij} is calculated as represented below in the equation 3.17.

$$(a_{ij}) = (b_{ij})^{-1} = \frac{adj\,(b_{ij})}{\det\,(b_{ij})} \tag{3.17}$$

where, $adj\,(b_{ij})$ is the transpose of the cofactor matrix of (b_{ij}) and the determinant of the transformation matrix is the *Jacobian* and is denoted by J. Thus, the inverse of the

transformation matrix can be rewritten as

$$(b_{ij})^{-1} = \frac{1}{J}\begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \beta_{31} & \beta_{32} & \beta_{33} \end{pmatrix} \tag{3.18}$$

where the coefficients are calculated as

$$\beta_{11} = \frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta} = \frac{\partial \xi}{\partial x}J \tag{3.19}$$

$$\beta_{12} = \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta} = \frac{\partial \xi}{\partial y}J \tag{3.20}$$

$$\beta_{13} = \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta} = \frac{\partial \xi}{\partial z}J \tag{3.21}$$

$$\beta_{21} = \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \zeta} = \frac{\partial \eta}{\partial x}J \tag{3.22}$$

$$\beta_{22} = \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \xi} = \frac{\partial \eta}{\partial y}J \tag{3.23}$$

$$\beta_{23} = \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \zeta} = \frac{\partial \eta}{\partial z}J \tag{3.24}$$

$$\beta_{31} = \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \xi} = \frac{\partial \zeta}{\partial x}J \tag{3.25}$$

$$\beta_{32} = \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \xi} - \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta} = \frac{\partial \zeta}{\partial y}J \tag{3.26}$$

$$\beta_{33} = \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} = \frac{\partial \zeta}{\partial z}J \tag{3.27}$$

The gradients of a scalar quantity along the Cartesian coordinates are transformed via the chain rule to the curvilinear coordinate system:

$$\frac{\partial \phi}{\partial x_i} = \frac{\partial \phi}{\partial \xi_j}\frac{\partial \xi_j}{\partial x_i} = \frac{\partial \phi}{\partial \xi_j}\frac{\beta_{ij}}{J} \tag{3.28}$$

The divergence of a vector quantity ϕ_i can be calculated as follows:

$$\frac{\partial \phi_i}{\partial x} = \frac{\partial \phi_i}{\partial \xi_j}\frac{\beta_{ij}}{J} \tag{3.29}$$

The Laplacian of a scalar quantity is represented in the computational space as defined below.

$$\frac{\partial}{\partial x_i}\left(\frac{\partial \phi}{\partial x_i}\right) = \frac{\partial}{\partial \xi_j}\left(\frac{\partial \phi}{\partial \xi_k}\frac{\beta_{ki}}{J}\frac{\beta_{ji}}{J}\right) \tag{3.30}$$

In a Cartesian coordinate or physical space, the general transport equation for an *in-*

compressible Newtonian fluid with a transport variable Φ is represented as,

$$\frac{\partial(\rho\Phi)}{\partial t} + \frac{\partial(\rho u_i \Phi)}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\Gamma_\Phi \frac{\partial \Phi}{\partial x_i}\right) + S_\Phi \tag{3.31}$$

where the source term S_Φ is represented as follows.

$$S_\Phi = -\frac{\partial p}{\partial x_i} + \rho g i \tag{3.32}$$

In the equation 3.31, the term Γ_Φ represents the diffusion coefficient of the transport equation. The transformed equation in curvilinear or computational space is represented based on the gradient, divergence and Laplacian as represented in the equations 3.28, 3.29 and 3.30 respectively.

$$\frac{\partial(\rho\Phi)}{\partial t} + \frac{1}{J}\frac{\partial(\rho U_j \Phi J)}{\partial \xi_j} = \frac{1}{J}\frac{\partial}{\partial \xi_j}\left(\frac{\Gamma_\Phi}{J}\frac{\partial \Phi}{\partial \xi_k} B_{kj}\right) + S_\Phi \tag{3.33}$$

The transformed source term in curvilinear coordinate system is represented as,

$$S_\Phi = -\frac{1}{J}\left(\frac{\partial p}{\partial \xi_k}\beta_{ki}\right) + \rho g i \tag{3.34}$$

The velocity component $U_j = (U, V, W)$ describes the *contravariant velocity* components which are based on the metric coefficients β_{ij}, the Jacobian J and the respective Cartesian velocity components. They can be evaluated as follows.

$$U_j = \frac{1}{J}\left(u\beta_{j1} + v\beta_{j2} + w\beta_{j3}\right) = \frac{1}{J}\left(u_n \beta_{jn}\right) \tag{3.35}$$

The coefficients B_{kj} are described as:

$$B_{kj} = \beta_{km}\beta_{jm} = \beta_{k1}\beta_{j1} + \beta_{k2}\beta_{j2} + \beta_{k3}\beta_{j3} \tag{3.36}$$

If the grid is orthogonal in the physical space, the off-diagonal elements β_{ij} in the transformation tensor vanishes and the general equation will be similar to the Cartesian form. The elements b_{ij} can be approximated at the control volume (CV) centers using second order central difference scheme.

$$(b_{ij})_P = \left(\frac{\partial x_i}{\partial \xi}\right)_P \approx \frac{x_{i,e} - x_{i,w}}{\delta\xi} \tag{3.37}$$

The distance between the centers of the control volumes is assumed to be 1, i.e. $\delta\xi = 1$. The j^{th} column elements of b_{ij} for the control volume represented in figure 3.2 can be represented as follows:

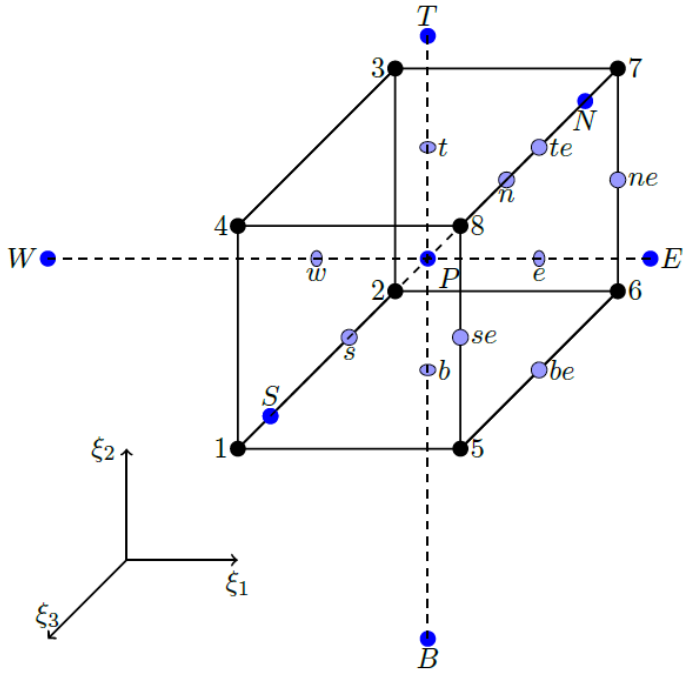


Figure 3.2: Representational control volume in the computational domain, picture taken from Münsch [25]

$$(b_{i1})_P = \left(\frac{\partial x_i}{\partial \xi}\right)_P \approx \frac{x_{i,5}+x_{i,6}+x_{i,7}+x_{i,8}}{4} - \frac{x_{i,1}+x_{i,2}+x_{i,3}+x_{i,4}}{4} \tag{3.38}$$

$$(b_{i2})_P = \left(\frac{\partial x_i}{\partial \eta}\right)_P \approx \frac{x_{i,3}+x_{i,4}+x_{i,7}+x_{i,8}}{4} - \frac{x_{i,1}+x_{i,2}+x_{i,5}+x_{i,6}}{4} \tag{3.39}$$

$$(b_{i3})_P = \left(\frac{\partial x_i}{\partial \zeta}\right)_P \approx \frac{x_{i,1}+x_{i,4}+x_{i,5}+x_{i,8}}{4} - \frac{x_{i,2}+x_{i,3}+x_{i,6}+x_{i,7}}{4} \tag{3.40}$$

The numerical treatment of the governing equation is done by Finite volume method. The discretization of the equation is presented in the following section.

3.2 Finite Volume discretization method

The transformed transport equation in the conservative form is represented in the following equation.

$$\int_{V_\xi} \frac{\partial \Phi}{\partial t} J dV_\xi + \int_{V_\xi} \frac{\partial \left(U_j \Phi\right)}{\partial \xi_j} dV_\xi = \int_{V_\xi} \frac{1}{J}\left(\Gamma_\Phi \frac{\partial \Phi}{\partial \xi_k} B_{kj}\right) dV_\xi + \int_{V_\xi} J S_\Phi dV_\xi \tag{3.41}$$

The control volume (CV) in the computational domain is related to a CV in physical

domain as given below.

$$dV_\xi = d\xi d\eta d\zeta = 1 \tag{3.42}$$

$$dV = dxdydz = Jd\xi d\eta d\zeta = J \tag{3.43}$$

Applying the *Gauss divergence* to the equation 3.41, the volume integral of the fluxes are transformed to the surface integral and the equation is rewritten as:

$$\int_{V_\xi} \frac{\partial \Phi}{\partial t} J dV_\xi + \int_{A_\xi} U_j \Phi dA_{\xi_j} = \int_{A_\xi} \frac{1}{J} \left(\Gamma_\Phi \frac{\partial \Phi}{\partial \xi_k} B_{kj} \right) dA_{\xi_j} + \int_{V_\xi} J S_\Phi dV_\xi \tag{3.44}$$

where, A_{ξ_j} is the area vectors of the control volume surfaces.

4 EVALUATION

5 Conclusion and Future Work

In this chapter we want to draw conclusions about the work, which has been done during this thesis.

5 BIBLIOGRAPHY

[1] P. Becker, S. R. Idelsohn, and E. Oñate. A unified monolithic approach for multi-fluid flows and fluid–structure interaction using the particle finite element method with fixed mesh. *Computational Mechanics*, 55(6):1091–1104, dec 2014. doi: 10.1007/s00466-014-1107-0.

[2] Alfred EJ Bogaers, Schalk Kok, B Dayanand Reddy, and Thierry Franz. Quasi-newton methods for implicit black-box fsi coupling. *Computer Methods in Applied Mechanics and Engineering*, 279:113–132, 2014.

[3] Michael Breuer, Guillaume De Nayer, Manuel Münsch, Thomas Gallinger, and Roland Wüchner. Fluid–structure interaction using a partitioned semi-implicit predictor–corrector coupling scheme for the application of large-eddy simulation. *Journal of Fluids and Structures*, 29:107–130, 2012.

[4] Maureen Caudill. Neural networks primer, part i. *AI expert*, 2(12):46–52, 1987.

[5] I Demirdžić and M Perić. Space conservation law in finite volume calculations of fluid flow. *International journal for numerical methods in fluids*, 8(9):1037–1050, 1988.

[6] Włodzisław Duch and Norbert Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2(1):163–212, 1999.

[7] Franz Durst. *Fluid mechanics: an introduction to the theory of fluid flows*. Springer Science & Business Media, 2008.

[8] Charbel Farhat and Michael Lesoinne. Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. *Computer methods in applied mechanics and engineering*, 182(3-4):499–515, 2000.

[9] Carlos A Felippa, KC Park, and Charbel Farhat. Partitioned analysis of coupled mechanical systems. *Computer methods in applied mechanics and engineering*, 190 (24-25):3247–3270, 2001.

[10] CC Feng. *The measurement of vortex induced effects in flow past stationary and oscillating circular and d-section cylinders*. PhD thesis, University of British Columbia, 1968.

[11] OM Griffin. Vortex-excited cross-flow vibrations of a single cylindrical tube. *Journal of Pressure Vessel Technology*, 102(2):158–166, 1980.

[12] OM Griffin. Vortex-induced vibrations of marine structures in uniform and sheared currents. In *NSF Workshop on Riser Dynamics*. University of Michigan, 1992.

[13] Gene Hou, Jin Wang, and Anita Layton. Numerical methods for fluid-structure interaction—a review. *Communications in Computational Physics*, 12(2):337–377, 2012.

[14] Björn Hübner, Elmar Walhorn, and Dieter Dinkler. A monolithic approach to fluid–structure interaction using space–time finite elements. *Computer methods in applied mechanics and engineering*, 193(23-26):2087–2104, 2004.

[15] Bruce M Irons and Robert C Tuck. A version of the aitken accelerator for computer iteration. *International Journal for Numerical Methods in Engineering*, 1(3):275–277, 1969.

[16] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

[18] Vivek Kumar. *Modeling and Numerical Simulations of Complex Transport Phenomena in Crystal Growth Processes: Modellierung und Numerische Simulationen Komplexer Transportvorgänge in Kristallzüchtungsprozessen*. Shaker, 2005.

[19] Ulrich Küttler and Wolfgang A Wall. Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72, 2008.

[20] Mouna Laroussi and Mohamed Djebbi. Vortex shedding for flow past circular cylinder: Effects of initial conditions.

[21] JC Muti Lin and Laura L Pauley. Low-reynolds-number separation on an airfoil. *AIAA journal*, 34(8):1570–1577, 1996.

[22] C Michler, SJ Hulshoff, EH Van Brummelen, and René De Borst. A monolithic approach to fluid–structure interaction. *Computers & fluids*, 33(5-6):839–848, 2004.

[23] Daniel P Mok and WA Wall. Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures. *Trends in computational structural mechanics*, 49:689–698, 2001.

[24] M Münsch and M Breuer. Numerical simulation of fluid–structure interaction using eddy–resolving schemes. In *Fluid Structure Interaction II*, pages 221–253. Springer, 2011.

[25] Manuel Münsch. Entwicklung und anwendung eines semi-impliziten kopplungsverfahrens zur numerischen berechnung der fluid-struktur-wechselwirkung in turbulenten strömungen mittels large-eddy simulationen. 2015.

[26] Michio Nishioka and Hiroshi Sato. Mechanism of determination of the shedding frequency of vortices behind a cylinder at low reynolds numbers. *Journal of Fluid Mechanics*, 89(1):49–60, 1978.

[27] Milovan Peric. A finite volume method for the prediction of three-dimensional fluid flow in complex ducts. 1985.

[28] Serge Piperno, Charbel Farhat, and Bernard Larrouturou. Partitioned procedures for the transient solution of coupled aroelastic problems part i: Model problem, theory and two-dimensional application. *Computer methods in applied mechanics and engineering*, 124(1-2):79–112, 1995.

[29] O Posdziech and R Grundmann. A systematic approach to the numerical calculation of fundamental quantities of the two-dimensional flow over a circular cylinder. *Journal of Fluids and Structures*, 23(3):479–499, 2007.

[30] Thomas Richter. A monolithic multigrid solver for 3d fluid-structure interaction problems. 2010.

[31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[32] Hermann Schlichting, Klaus Gersten, Egon Krause, Herbert Oertel, and Katherine Mayes. *Boundary-layer theory*, volume 7. Springer, 1955.

[33] Joe F Thompson, Zahir UA Warsi, and C Wayne Mastin. *Numerical grid generation: foundations and applications*, volume 45. North-holland Amsterdam, 1985.

[34] Jesus Gerardo Valdés, Eugenio Oñate, and J Miquel. Nonlinear analysis of orthotropic membrane and shell structures including fluid-structure interaction. 2016.

[35] EH Van Brummelen, SJ 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.

[36] Wolfgang A Wall, Daniel P Mok, and Ekkehard Ramm. Partitioned analysis approach of the transient coupled response of viscous fluids and flexible structures. In *Solids, structures and coupled problems in engineering, proceedings of the European conference on computational mechanics ECCM*, volume 99, 1999.

[37] CHK Williamson. 2-d and 3-d aspects of the wake of a cylinder, and their relation to wake computations. *Lectures of Appl. Maths., Am. Math. Soc.*, 28:719, 1991.

[38] Saim Yigit, DD Sternel, and M Schäfer. Efficiency of fluid-structure interaction simulations with adaptive underrelaxation and multigrid acceleration. *The International Journal of Multiphysics*, 1(1), 2007.

[39] CY Zhou, RMC So, and K Lam. Vortex-induced vibrations of an elastic circular cylinder. *Journal of Fluids and Structures*, 13(2):165–189, 1999.