Shape Sensitivity Analysis for Coupled Fluid-Solid Interaction Problems

Koorosh Gobal

January 18, 2016

Contents

List of Figures

List of Tables

Abstract

In this paper, a robust continuum sensitivity formulation for the shape sensitivity analysis of weakly coupled aero-structural systems is derived. In this method, the solid boundaries are modelled using the immersed boundary approach. This simplifies the grid generation for the complex and deforming geometries since the computational mesh does not need to conform to the boundaries. The sensitivity analysis consists of differentiating the continuum form of the governing equations where the effect of the solid boundaries are modelled as additional forcing terms in these equations. By differentiating the governing equations, it is possible to reuse the operators utilized for solving the governing equations. Therefore, there is no need to develop new solvers for the solution of sensitivity response. This methodology is applied to different demonstration problems including flow over a cylinder and a simplified aeroelastic model of a wing. The wing structure is modelled as a beam with the lifting surface mounted at the tip where the load is transfered to the structure through the mounting point. The sensitivity results with this approach compares well with the complex step method results. Moreover, it is shown that the methodology is capable of handing complex shapes with high Reynolds numbers.

Chapter 1

Introduction

1.1 Motivation

Fluid-structure interaction (FSI) problems play an important role in many scientific and engineering fields, such as automotive, aerospace, and biomedical industry. Despite the wide application, a comprehensive study of FSI systems still remains a challenge due to their strong nonlinearity and multi-physics nature. For most FSI problems, analytical solution of responses are not available, and physical experiments are limited in scope, expensive to conduct, and time consuming. Therefore, in order to get more insight in the physics involved in the complex interaction between fluids and solids, numerical simulations are typically used. The numerical solutions are conducted using Computational Fluid Dynamics (CFD) models for the flow field and Finite Element Analysis (FEA) for the structural response. Nevertheless, the prohibitive amount of computations has been one of the major issues in the design optimization of such coupled multidisciplinary systems. The other bottle neck is generating an appropriate computational domain that represents the fluid and solid regions with desired fidelity. The effort and time required to take a geometry from a CAD package, refining the model and generating a computational mesh is usually a large portion of the overall human time required for the simulation. This cannot be automated for complex and moving domains. Therefore, techniques that decouple the mesh topology from the shape of the solid boundary can drastically simplify the simulation of flow over complex bodies. The Immersed Boundary (IB) method addresses these challenges for complex body shapes by introducing a new approach to define the solid boundaries.

In aerospace and automotive systems, configurations are optimized for achieving maximum performance and design targets. Due to the large amount of computations involved in the FSI simulation, the gradient based methods are the best candidates for design optimization of such problems. Sensitivity analysis is the integral part of gradient based methods. Although there are analytical techniques for efficient and accurate sensitivity calculation, they have not been implemented in commercial CFD packages. Therefore, most gradient optimization techniques depend on finite difference method for sensitivity calculation when solving FSI problem. It is well known that finite difference suffers from low accuracy and high cost.

The motivation for the research proposed in this document is in two areas. First, we want to have sensitivity analysis capabilities that can treat the CFD solvers as black-box. This means that we can solve both the governing equations and the sensitivity response using the same code. Second, a robust analysis technique for the coupled FSI system based on IB method is formulated. The current approach of IB is not suited for the sensitivity analysis due to the discontinuities in its formulation. This will be explained in more detail in the following Chapters.

1.2 Literature Review

1.2.1 Sensitivity Analysis

Sensitivity analysis consists of computing the derivatives of the solution of the governing equations, i.e. the sensitivity of displacement, velocity, or pressure with respect to one of the several independent design variables, i.e. shape of boundaries or cross-sectional and size of elements. There are various applications for sensitivity information such as aircraft trajectory optimization [1], improving the accuracy of surrogate models as in gradient enhanced Kriging [2] or quantification of uncertainty [3]. However, our main motivation is the use of this information in gradient-based optimization. The calculation of gradients is often the most expensive step in the optimization cycle therefore, efficient methods for accurate calculation of sensitivities are vital to the optimization process. As shown in Figure 1.1, methods for sensitivity calculation can be put into three major categories: i) numerical, ii) analytical, and iii) automatic differentiation.

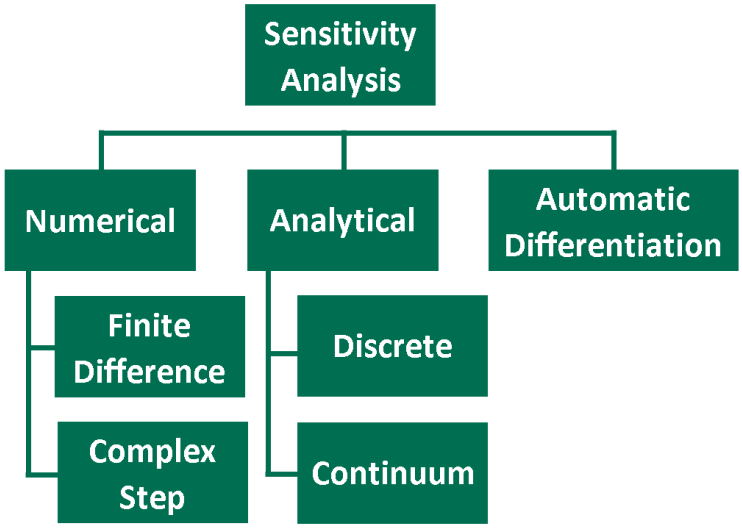


Figure 1.1: Sensitivity calculation techniques.

The Finite Difference (FD) method is probably the easiest method to implement using a commercial software for calculating the sensitivity of a variable. The fact that they can be implemented even when a given computational model is treated as a black box makes most gradient based optimzation algorithms perform finite differences by default when the user does not provide the required gradients. However, the computational cost associated with finite difference scheme for large systems can become very high. For a system with n number of design variables, the analysis has to be performed $n+1$ times to calculate the design sensitivities. Furthermore, to ensure the accuracy, convergence study needs to be done for selecting the appropriate step size for the finite difference. The inaccuracy of finite differencing could result in convergence difficulties and inaccurate optimum results [4].

Complex Step (CS) method avoids the loss of precision in finite differences approximation by employing complex arithmetic [5]. The complex step derivative is defined as shown in Equation (1.1).

$$\mathcal{F}'(u;b) = \frac{\mathrm{Im}\left[\mathcal{F}(u;b+ih)\right]}{h} \tag{1.1}$$

where $\mathcal{F}(u;b)$ is the function of interest such as the airfoil lift that depends of a response variable, u i.e. free-stream velocity, and design variable, b i.e. shape of the airfoil. Complex step approximates the sensitivities by perturbing the design variable by an imaginary value of ih and then evaluating the imaginary portion of the resulting response. Using the complex step method, we can choose a small step size for h without loosing accuracy due to no subtraction

involved in CS formulation. However, many commercial packages such as ANSYS or NASTRAN cannot handle complex arithmetic which makes the implementation of complex step method infeasible. Moreover, the high cost of finite difference is still associated with the complex method as well.

Automatic differentiation (AD) is based on the systematic application of the differentiation on computer programs [6]. In the AD approach, the chain rule of differentiation is applied to every line in the program assuming that the computer program consists of a sequence of explicit functions that act successively on some variables. Therefore, by differentiating each of these functions and applying the chain rule, it is possible to calculate the sensitivities.

There has been extensive research on utilizing AD for optimization. Bischof et al. used AD for calculating the sensitivities using a CFD solver. They used ADIFOR for differentiating the source code of their CFD code (TLNS3D) which later used for calculating the sensitivity of a transonic flow to change in the boundary conditions. Hascout et al., also used AD for a sonic boom reduction under a supersonic aircraft. In all of these works in order to implement AD, it is needed to have access to the source code and modification of the solver extensively to calculate the sensitivities. This make the use of this method for commercial codes impractical since the source code is usually not available.

The short comings of numerical and AD techniques, demands for more sophisticated methods for sensitivity calculation. These techniques are generally known as analytical methods. Formulation of the analytical sensitivities requires derivation of analytic sensitivity equations. These are obtained by differentiating the governing equations with respect to design variables such as shape of the boundaries. Analytical methods can be further categorized based on how the sensitivity equations are derived and solved. Typically the continuum equations of the system are solved using an approximate method which discretizes the governing equations. The Discrete Sensitivity Analysis (DSA) technique, differentiates the discretized governing equations with respect to design variables to get the analytical sensitivity equations [7]. The Continuum Sensitivity Analysis (CSA) differentiates the continuum equations for formulating the sensitivity equations. This system of equations is later solved for analytical sensitivities.

Regardless of continuum or discrete approach for deriving the sensitivity equations, these can be solved using two approaches: i) the direct method and ii) the adjoint method. This

need to be explained before talking about the sensitivity calculating techniques in detail. To better explain these two method, lets use the finite element formulation for the displacement method of analysis.

$$KU = F \tag{1.2}$$

where K is the stiffness matrix that represents a structure, U is the displacement vector, and F is the load vector. The sensitivity of a response, R, (e.g. stress, displacement) with respect to a design variable b_i is determined by using chain rule for differentiation as shown in Equation (1.3).

$$\frac{dR}{db_i} = \frac{\partial R}{\partial b_i} + \frac{\partial R}{\partial U}\frac{\partial U}{\partial b_i} \tag{1.3}$$

The best way to calculate the displacement sensitivity in Equation (1.3), is to differentiate the governing Equation of (1.2) with respect to design variable b_i.

$$\frac{\partial K}{\partial b_i}U + K\frac{\partial U}{\partial b_i} = \frac{\partial F}{\partial b_i}$$

Above equation is rewritten as

$$\frac{\partial U}{\partial b_i} = K^{-1}\underbrace{\left[\frac{\partial F}{\partial b_i} - \frac{\partial K}{\partial b_i}U\right]}_{\text{pseudo-loads}} \tag{1.4}$$

The response sensitivity, dR/db_i, is calculated by substituting Equation (1.4) in (1.3). The *direct* method first calculates the displacement sensitivity, $\dfrac{\partial U}{\partial b_i}$ and uses that to calculate $\dfrac{\partial R}{\partial U}\dfrac{\partial U}{\partial b_i}$ to form the response sensitivity. This method is performed ones for each design variable. The *adjoint* method first calculates $[K^{-1}]^T\left(\dfrac{\partial R}{\partial U}\right)^T$ and then dot product it with the pseudo-load to form the second part of the response derivative. The method is performed one for each response. Therefore, if the number of responses are smaller than the design variables the adjoint method will have better performance in terms of simulation cost [8].

The discrete sensitivity analysis has been historically the method of choice to calculate the high-accurate sensitivity when the details of analysis such as discretization approach and matrix assembly of discretized equations are available [9]. This method has been adopted by

the structural optimization community, and been applied to fluid-solid interaction problems as well. Reuther et al., used the discrete method for aerodynamic shape optimization of a complex aircraft configuration [10]. They used Euler flow as the aerodynamics theory where they optimized different configurations for transonic and supersonic regimes where they only focused on the fluid pressure. Martins et al., developed an adjoint method for sensitivity analysis for an aero-structural aircraft design framework where the sensitivities were computed using a coupled adjoint approach. The framework was used on a supersonic business jet to calculate the sensitivity of drag with respect to the Outer Mold Line (OML) of the aircraft. They updated the shape of the wing airfoils by using Hicks-Henne functions. These airfoils were then linearly lofted to generate the wing. The advantage of Hicks-Henne functions is that when they are applied to a smooth airfoil, it remains smooth. In their work, the discretization details of the solver needs to be known to calculate the sensitivities since they used DSA [11]. Newman et al., used this approach for aerodynamic shape sensitivity analysis and design optimization of geometrically complex configurations [12]. In their work, the flow is modelled using nonlinear Euler equations where the fluid domain is discretized using unstructured grid. They used highly efficient Gauss-Seidel and GMRES solvers for the solution of linear aerodynamics equations. The mesh deformation due to change in the optimization loop was performed by considering the mesh as a system of interconnected springs. They had to calculate the grid sensitivities by differentiating the grid adaptation algorithms which adds to the cost of the sensitivity analysis. Moreover, mesh deformation algorithms do not always result in usable domain for solving the governing equations. For cases where the boundary deformation is large, mesh deformation fails due to tangled meshes or negative cell volumes [13]. Jameson et al., also used unstructured grid for aerodynamic shape optimization [14]. They used adjoint formulation to calculate the aerodynamic shape sensitivities of a complete aircraft configuration. In their work the flow is modelled using Euler equation and the mesh deformation is performed using the spring method. Using DSA they were able to calculate the sensitivities of pressure coefficient of the aircraft wing with respect to its shape and use this data to optimize the shape of the wing. However, their method cannot be implemented in a commercial solver without access to the source code. Moreover, mesh deformation is still used in this approach that reduces the robustness of their technique. As a matter of fact, source code modification is essential in all

other related research papers published in the area as well [15, 16, 17, 18]. However, the source code is usually inaccessible and very complex. Therefore, there is a great interest in sensitivity calculation techniques which require minimum knowledge of the analysis code. This can be achieved using the sensitivity formulation that operates on the governing equations before they are discretized. These method are commonly known as Continuum Sensitivity Analysis (CSA) Methods [19].

The CSA, involves solving a set of partial differential equations named the Continuum Sensitivity Equations (CSEs) to get the analytical sensitivities. When deriving the CSEs, the governing equations can either be differentiated in local or total form. The local formulation only reflects the effect of design variable on change on the response at specific point in space. On the other hand, the total formulation of sensitivities takes the movement of physical points due to change in the shape of the domain into account.

Choosing between local and total differentiation depends on the Lagrangian or Eulerian representation of the governing equations. In the general case of continuum mechanics, displacement and velocity are vectors. In any application, we have the choice of writing these vectors as functions of the position of material particles before deformation. This is called the Lagrangian description of motion and is really helpful for visualizing the deformations. This technique is mostly adopted in solid mechanics where we track the material points as the deformations are usually assumed to be small. However, in the fluid flow problems, since it is generally hard to identify a reference configuration and the deformations are large, it is preferable to write the displacement and velocities as functions of the deformed position of the particles. These quantities are now defined for a particular point in space that does not move with the particles. This is called the Eulerian description of motion. As CSA was matured over the years, the computational fluids community adopted the local CSEs, since its formulation is consistent with the Eulerian formulation of the governing equations [20, 21]. The structural optimization community adopted the total formulation for the sensitivities since its formulation was consistent with the Lagrangian formulation of structural mechanics. Nevertheless, the total and local formulations for the sensitivities can be converted from one form to the other. This will be explained in more details in next chapter.

In summary, sensitivity analysis in general is more matured for structural problems than

for fluid dynamics. Nevertheless, neither of these disciplines typically employ CSA for the sensitivity calculation. The main reason for this is the need for higher derivatives on the boundaries that can be hard to approximate [22]. Aurora and Haug [23], followed by Dems and Mroz [24], were among the first to develop the concept of CSA for structural optimization. They modeled the sensitivities as functionals therefore, they were able to convert the sensitivity integrals over the entire domain to the integrals over the boundaries. Although using this approach it is possible to reduce the cost of the simulation, accurate values of functionals are required at the boundaries. This is not always achievable especially for finite element analysis where the solution accuracy drops near the boundaries. The lack of applicability of CSA in structural optimization is due to the complicated definition of the boundary conditions and maturity of discrete methods for the sensitivity calculation. In recent years, Cross and Canfield [22] developed specific local CSA formulation to handle these issues. They used Spatial Gradient Reconstruction (SGR) technique to approximate the gradient with higher order accuracy near the boundaries. This is essential for calculating accurate sensitivities.

The first application of CSA in optimization problems for fluid dynamics is the work by Borggaard and Burns [25] for shape sensitivity analysis of inviscid supersonic flows over rigid bodies. Stanley and Stewart [26] applied CSA in a fluid mechanics discipline with a goal for aerodynamic design. Pelletier and Etienne have applied CSA to numerous fluid-structure interaction (FSI) problems [27] focused mainly on sensitivities of fluid flow parameters near the structure. Liu and Canfield have employed CSA for shape optimization of nonlinear structures subject to an aeroelastic gust response [28]. They used the finite element method to solve the potential flow around an airfoil and applied CSA to find the airfoil pressure coefficient sensitivity with respect to the maximum camber.

In almost all of the research done on sensitivity calculation of flow field to shape design variable, body conformal grids were used. The conforming mesh methods consider the interface conditions as physical boundary conditions, which treat the interface location as part of the solution and requires meshes that conform to the interface. Using this approach it is possible to represent the solid boundary shape with good accuracy. However, due to the coupling of fluid mesh topology and solid boundary shape, with the movement and/or deformation of the solid structure, re-meshing (or mesh-updating) is needed. Although conforming mesh methods

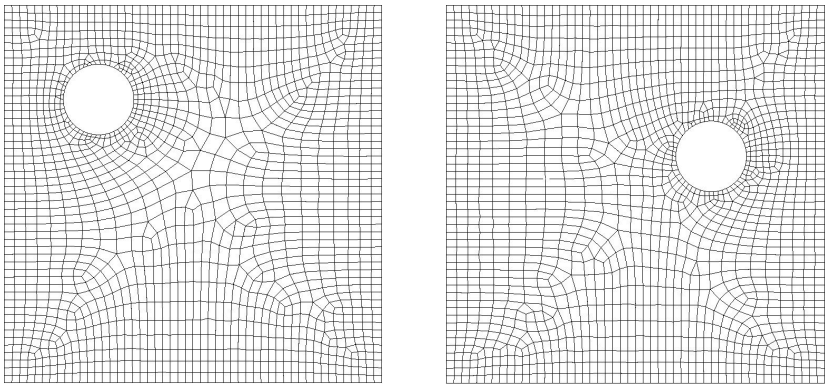
have been widely used in many FSI problems, they are cumbersome, if not impossible, to apply to problems with large deformations [29]. The other shortcoming of body-conformal grids, is the effect of mesh deformation on the sensitivity analysis. Since the computational domain is affected by change in the shape of the boundaries, it is required to calculate the mesh sensitivities as well [30]. This adds to the computational effort for calculating sensitivities.

The shortcoming of a robust grid generation and the additional cost of calculating mesh sensitivities motivated an important research effort to develop a technique that does not require fluid domain's mesh modification throughout the optimization iterations. One of the possible candidates to achieve this goal, is the use of Immersed Boundary (IB) methods to decouple the fluid mesh from the shape of solid domain.

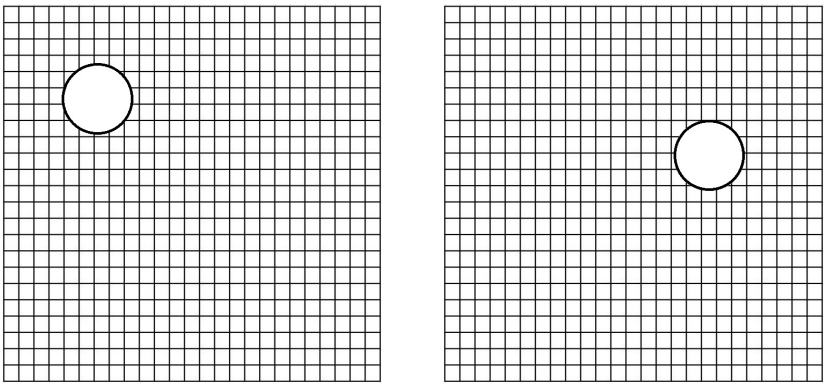
1.2.2 Immersed Boundary Method

Traditionally the analysis techniques for flow over complex bodies are based on body-fitted multi-block or unstructured grid methods. However, in the last decade another class of techniques, called the immersed boundary methods, have been introduced. The term immersed boundary is first used by Peskin [31] to simulate the blood flow through heart valves. What distinguishes this method from the other methods for representing the solid boundaries, is that the flow is solved on the Cartesian grid that does not necessarily conforms to the boundaries. Therefore, the mesh generation is greatly simplified and in case of moving/deforming boundaries, there is no need to update the mesh during the simulation. A separate formulation is used to impose the effect of the boundaries on the governing equations.

Consider the simulation of flow past a circular cylinder as shown in Figure 1.2. Generating structured or unstructured grids is achieved in two steps. First, a surface grid covering the boundaries is generated. This is then used as a boundary condition to generate a grid in the volume occupied by the fluid. The differential form of the governing equations is then transformed to a curvilinear coordinate system aligned with the grid lines [32]. The governing equations are then discretized on this curvilinear grid and solved using appropriate technique.



(a) Conforming mesh, $t = t_1$ (b) Conforming mesh, $t = t_2$



(c) Nonconforming mesh, $t = t_1$ (d) Nonconforming mesh, $t = t_2$

Figure 1.2: Example of conforming and nonconforming meshes.

Non-body conformal Cartesian grid can also be utilized for this simulation as shown in Figure 1.2. In this approach the IB would still be represented through some means such as a surface grid, but the Cartesian volume grid would be generated with no regard to this surface grid. Thus, the solid boundary would cut through this Cartesian volume grid. Because the grid does not conform to the solid boundary, incorporating the boundary conditions would require modifying the equations in the vicinity of the boundary. Assuming that such a procedure is available, the governing equations would then be discretized using a finite-difference, finite-volume, or a finite-element technique without resorting to coordinate transformation or complex discretization operators.

Depending on how the effect of solid boundaries are imposed, IB methods can be divided into three categories: i) Continuous forcing, ii) discrete forcing, and iii) cut-cell method.

The continuous forcing method was originally used by Peskin [31] and later further developed by others researchers [33, 34, 35]. In this approach, the boundary configuration is described by a curve $x(s, t)$ (Lagrangian nodes) where the location of each point on this boundary is governed

by its equations of motion. The forces that the curve $x(s,t)$ exerts on the fluid is calculated by the constitutive law of the curve $x(s,t)$ that relates the displacements to stress values. These stress values are transferred to the Navier-Stokes (NS) (Eulerian nodes) for the fluid by means of a Dirac delta function. Practical implimentation of this method rests in representing the Dirac delta function as a discrete function that has the same properties. This method is applied to a variety of problems, including Cardiac blood flow [36], animal locomotion [37], multiphase flows [38], and particle sedimentation [39].

Peskin's method is well suited for the elastic boundaries. For stiff boundaries, the constitutive laws of the solid boundaries will result in instabilities in the solution of governing equations [40]. The virtual boundary method of Goldstein [41] enables us to handle these rigid boundaries. The main idea of this approach is the same as Peskin's method where the solid boundary is treated as a virtually existing surface embedded in the fluid. The force on this surface is calculated by the requirement that the fluid velocity should satisfy the no-slip condition. Since this body force is not known a priori, it is calculated in a feedback way.

The penalization method is also categorized under the continuous forcing approaches to the IB problem. This technique is based on the Brinkman equation that describes the fluid flow through a porous medium. The Brinkman equation is analogous to Fourier's laws in heat conduction and Ohm's law in the field of electrical engineering [42]. This approach was first proposed by Arquis and Caltagirone [43] where they imposed the boundary conditions by adding the penalization terms to the momentum equations. The main idea of this approach is to model the solid obstacles as a porous medium with the porosity of $\mathcal{K}$. The porosity is a measure of the void spaces in a material. Therefore, by selecting low values for porosity, the porous domain acts as a solid wall. It has been demonstrated that the solution of the penalized incompressible NS equations converges to the exact solution as the porosity approaches zero [44] however in the practical implementation, extreme low values for porosity will result in systems with a very large stiffness and numerically unstable. To apply the porosity within the solid domain, a Heaviside function is used by different researchers [45, 46].

The continuous forcing approach is very attractive for flows for both rigid and elastic boundaries due to its ease of implementation. This technique can be added to an already available commercial package such as ANSYS Fluent for CFX with limited effort. One of the shortcom-

ings of this approach is the smoothing of the forcing function near the boundary. This can result in inability for sharp representation of the immersed boundary in cases such as compressible flows. However, accuracy for representing the boundaries can be improved by refining the mesh and using higher order force mapping schemes.

Mohd-Yusof formulated the discrete formulation of IB method [47] for addressing the time penalties associated with the simulation involving moving boundaries. The main idea of this technique is same as the virtual boundary method where a force term is added to the momentum equations to represent the solid boundary. However, in the current approach the momentum equation manipulation is done in the discrete manner. In the work of Mohd-Yusof [47] and Verzicco et al. [48] this is done by modifying the discretization stencil in the vicinity of the boundary curve so that the velocity values on this boundary satisfy a pre-defined condition. The major advantage of this approach is the absence of user specified parameters however, its implementation depends strongly on the discretization approach used for the analysis. Therefore, it cannot be easily implemented in commercial codes. This implementation of discrete forcing method has been applied to many different problems such as turbulent flow inside an internal combustion engine [48], flow past 3D bluff bodies [49], and flow in a cylindrical stirred tank [50].

In the cut-cell methods, the grid cells are cut by the solid boundary and reshaped so that they form a boundary-conforming, unstructured grid. This is shown in Figure 1.3.

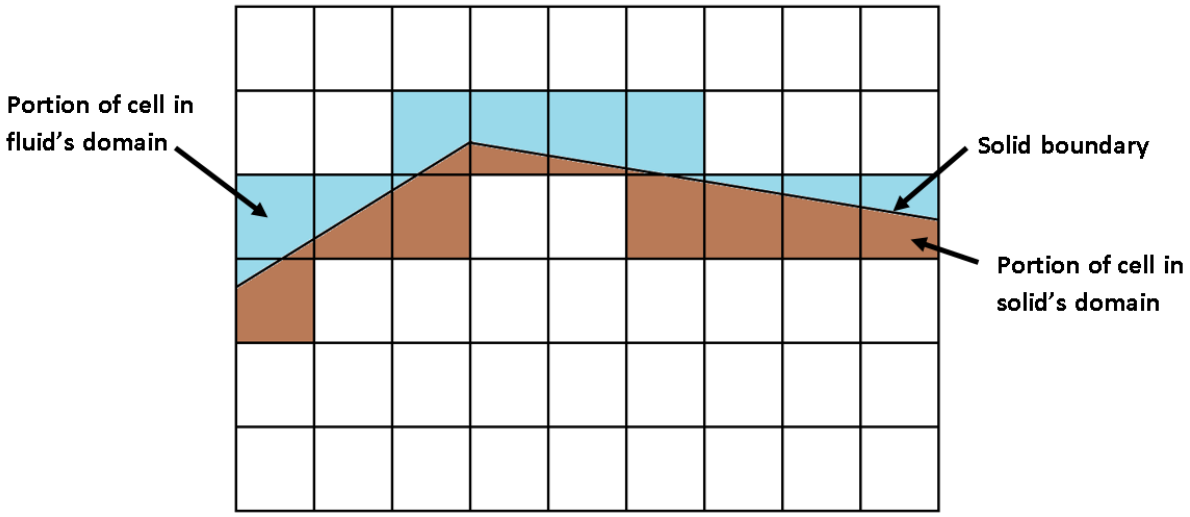


Figure 1.3: Modified mesh near the solid boundary for cut-cell method.

The cut-cell method was first introduced by Clarke [51] for inviscid flows. This method

has been applied to both collocated and staggered grids [52] however, most of the applications were focused on 2D flows [53, 54]. This is because of the many possibilities for the geometrical shape of the cut-cell, that makes the flux calculation and therefore numerical implementation extremely difficult.

1.2.3 Sensitivity analysis for IB method

The body of the research done in the immersed boundary community has been concentrated on the improvement of the method accuracy and resolving the stability issues. However, there has not been much effort in the sensitivity calculation using an IB formulation. There has been some research on the sensitivity analysis using the penalization framework. Borrvall and Petersson applied the penalization method for topology optimization of fluids in Stokes flow [55]. They used the discrete sensitivity analysis to calculate the sensitivity of fluid pressure to the shape of the boundaries. They used this technique to optimize the shape of a diffuser and also a pipe bend. They verified their methodology for outflow problems as well, where they optimized the shape of solid domain to maintain the least possible pressure drop by constraining the penalized region volume. Challis and Guest investigated the level set formulation for the topology optimization of fluids in Stokes flow [56]. They used the penalization technique to formulate the no-slip condition on the solid boundaries. They implemented the topological sensitivities in their solver where they used power dissipation minimization to optimize the shape of a diffuser and a connecting 3D pipe. The penalization method is probably the easiest of the IB methods to implement. This is because it does not have a free parameter such as in virtual boundary method and does not depend on the discretization such as discrete forcing methods. Moreover, no interpolation is needed for applying penalization method to a computational domain. In the penalization method, the fluid and solid domains are differentiated based on a scalar function (porosity) which is very similar to the density based approaches in the topology optimization community [57]. This is probably the biggest reason for researchers to use penalization technique for shape optimization using IB method since the available techniques from topology optimization community can be used [58, 59]. Similar to topology optimization, the shortcoming of penalization technique is in its accuracy for representing the boundaries. Since the nodes are assigned with the porosity values, it is extremly difficult if not impossible to

control the exact location of immersed boundary if it does not coincide with the computational nodes. Moreover, the current demonstration problems for the penalization technique are only applicable to low Reynolds numbers flow.

1.3 Research Contribution

Design optimization for coupled fluid-solid interaction problems is a challenging effort. The current state-of-art technique handles the MDO problem by using unstructured gird and using DSA for sensitivity calculation. However, these techniques cannot be used along with arbitrary software package without a deep underestaning of how the governing equations are formulated and discretized within the solver. Moreover, due to the use of body-conformal grids for fluid domain discretization, additional cost and effort are introduced in both the analysis and sensitivity calculation. Approximating the mesh sensitivities and mesh deformation are two examples for this extra calculation. This research presents a specific formulation for immersed boundary method along the continuum sensitivity analysis for satisfying both of the issues regarding mesh deformation and sensitivity implementation. The specific contributions of this research are as follows

- Two different classes of continuous immersed boundary methods are explained and used within CSA framework to calculate the sensitivity of different flow parameters with respect to shape design variables. The immersed boundary method are based on relaxed Dirac delta functions to transfer the data between the fluid and structure domains however, this formulation cannot be used within the CSA framwork. Smooth representation for Dirac delta function is developed for this need.

- This research is the first application of CSA for sensitivity analysis of full Navier-Stokes equations. The previous works mainly focoused on Euler and Potential flows where the viscous shear effects are ignored. Using full Navier-Stokes equation, it is possible to reach higher fidelities in design optimization for FSI problems.

- The use of continuum immersed boundary methods for sensitivity analysis of flows with moderate to high Reynolds number is another contribution of this work. Most of the work

done in this area regarding the sensitivity analysis is based on flows with lower Reynolds number such as Stokes flows.

Chapter 2

Design Sensitivity Analysis

In this chapter, the concept of discrete and continuum sensitivity formulation is discussed, and the general approach for deriving the sensitivity equations is presented. Then, the difference between the local and total formulation of the sensitivity response is discussed and finally, the independence of continuum sensitivity analysis to discretization method is proven. This enables us to reuse the solver of governing equations to calculate the sensitivity response. Discrete sensitivity formulation is not capable of using the same solver for both that solution of governing equation and sensitivity analysis. The two sensitivity analysis techniques are applied to a heat transfer benchmark problem where the sensitivity of response to the shape of the domain is calculated. This problem is also used in the next chapter for implementation of different immersed boundary methods.

2.1 General Formulation

The general shape for the computational domain is defined in Figure 2.1. The response variable on this domain can represent a fluid, i.e. pressure or velocity, or a solid, i.e. displacements. Nevertheless, in any of these cases, the response is calculated by solving a governing equation subjected to boundary conditions.

In this research, the governing equation and the boundary conditions are represented in the functional form as shown in Equation (2.1).

$$\mathbf{A}(\mathbf{u}, t; \mathbf{b}) = \mathbf{f}(\mathbf{x}, t; \mathbf{b}) \quad \text{on} \quad \Omega \tag{2.1a}$$

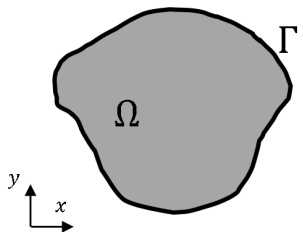


Figure 2.1: General computational domain Ω with boundary Γ.

$$\mathbf{B}(\mathbf{u}, t; \mathbf{b}) = \mathbf{g}(\mathbf{x}, t; \mathbf{b}) \quad \text{on} \quad \Gamma \tag{2.1b}$$

where $\mathbf{A}$ is the governing equation, such as Navier-Stokes equations for the fluid or elastic equations for the solid domain, and $\mathbf{B}$ is the boundary condition. $\mathbf{u}$ is the response variable such as displacement or pressure. t is time, $\mathbf{b}$ is the design variable such as shape or size, and $\mathbf{x}$ is the spatial coordinate. $\mathbf{f}$ and $\mathbf{g}$ are the values of the governing equation and boundary conditions. It should be noted that in this formulation, the functions $\mathbf{A}$ and $\mathbf{B}$ are only implicitly dependent on the design variable $\mathbf{b}$. This is important when deriving the sensitivity equations later in this chapter. This implicit dependence is represented by using the semicolon symbol in the definition of $\mathbf{A}$ and $\mathbf{B}$.

For the general case of problem formulation, the total sensitivity of response variable, $\mathbf{u}$, with respect to the i-th design variable, b_i, is written as:

$$\frac{D\mathbf{u}}{Db_i} = \underbrace{\frac{\partial \mathbf{u}}{\partial b_i}}_{\text{local derivative}} + \underbrace{\frac{\partial \mathbf{u}}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial b_i}}_{\text{convective term}} \tag{2.2}$$

The total derivative is known as the material derivative in continuum mechanics [60]. This total sensitivity defines the change of response variable, $\mathbf{u}$, to design variable and space dependent changes. The material derivative consists of the local derivative, $\partial \mathbf{u}/\partial b_i$, plus the convective term, $\partial \mathbf{u}/\partial \mathbf{x} \cdot \partial \mathbf{x}/\partial b_i$. The local derivative is the measure of change in the response variable due to change in the design parameter. Whereas, the convective term accounts for the movement of points in space due to change in the design variable. This is especially applicable to shape sensitivity analysis where the change in design variable will cause the material points to move [22].

The convective term consists of two separate gradients: i) $\partial \mathbf{u}/\partial \mathbf{x}$ which represents the spatial gradient of the response variable, and ii) $\partial \mathbf{x}/\partial b_i$ that defines the sensitivity of the location of

computational nodes with respect to changes in the design variable. The response gradient, $\partial u/\partial x$, is calculated from the analysis results using finite difference approach or derivative of shape functions in FEA formulation. The calculation of domain sensitivity, $\partial \mathbf{x}/\partial b_i$, requires more attention.

A standard approach for calculating the domain sensitivity is to use the techniques used to deform the body-conformal mesh in CFD/FEA simulations. These methods are commonly based on representing the computational grid as a system of springs that connected to each other at the nodes. This system is modeled and solved using structural analysis techniques, where the sensitivities can be easily implemented. This is effectively a shape sensitivity analysis for the structural problem [61]. This step is removed from the analysis if the computational domain is not affected by the design variable since $\partial x/\partial b$ is equal to zero [62]. This is one of the reasons to use the immersed boundary calculation since it reduces the cost of the simulation. This is discussed in more details in Chapter 3.

2.2 Benchmark Cases

To compare the discrete and continuum sensitivity analysis formulations, two problems from different disciplines are selected. As a simplified representation for the fluid/thermal systems, the one-dimensional heat transfer analysis in a beam is selected. This problem is used by different researchers for investigating sensitivity analysis for nonlinear systems [63], thermal design, and monitoring [64, 65]. The special interest in this problem is due to availability of analytical solution since this can be used to verify the numerical results of the sensitivity analysis. For the solid mechanics problem, a finite element model of an axial bar with distributed load is selected [66]. This problem is used by different researchers as a demonstration case for sensitivity analysis [22, 67]. These benchmark cases are explained in more details in the following sections.

2.2.1 Heat transfer benchmark case

The temperature in the one-dimensional domain is governed by the Laplace equation as shown in Equation (2.4).

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.3}$$

Where T is the temperature, x is spatial variable, t is time, and α is the thermal diffusivity. For this problem we are only interested in the steady state solution of the system; therefore, the right-hand-side of Equation (2.3) is equal to zero. Thus, the governing equation for this problem is written as:

$$\frac{\partial^2 T}{\partial x^2} = 0 \tag{2.4}$$

The boundary conditions are defined as constant temperatures at the two ends of the domain as T_0 and T_L. The domain length is selected as L. This is shown in Figure 2.2.

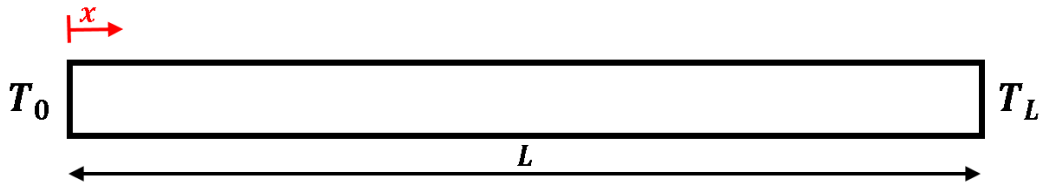


Figure 2.2: One dimensional domain with heat conduction.

The analytical solution for this problem is available and is written as shown in Equation (2.5).

$$T = \frac{T_L - T_0}{L}x + T_0 \tag{2.5}$$

The analytical sensitivity of the temperature with respect to the beam's length is calculated by differentiating Equation (2.5) to L.

$$\frac{\partial T}{\partial L} = -\frac{T_L - T_0}{L^2}x \tag{2.6}$$

This is later used for verifying the discrete and continuous sensitivity results.

2.2.2 Solid mechanics benchmark case

The axial bar is shown in Figure 2.3. The length of the bar is selected as $L = 1$ with axial stiffness of $EA = 1$. The distributed load is selected as a sine wave, $F_d(x) = sin(\pi x/L)$, with an extra axial load at the end, $F_p = 1/\pi$. The beam is fixed using a spring of stiffness, $k = 10$, at $x = L$.

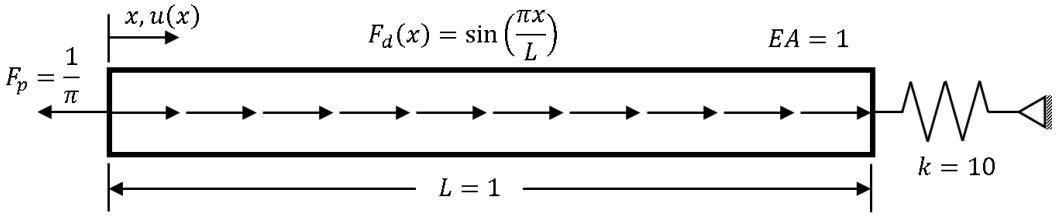


Figure 2.3: Axial bar under distributed loading.

The governing equation for the displacement of the axial bar and its corresponding boundary conditions are defined in Equation (2.2.2). The bar length is selected as the design variable; therefore, it is included explicitly in the governing equation and boundary condition definitions.

$$\frac{\partial^2 u}{\partial x^2} + \sin\left(\frac{\pi}{L}x\right) = 0 \tag{2.7a}$$

$$\begin{cases} \left.\dfrac{\partial u}{\partial x}\right|_{x=0} = \dfrac{1}{\pi} \\ \left.\dfrac{\partial u}{\partial x}\right|_{x=L} = -10u(L) \end{cases} \tag{2.7b}$$

For this problem, the analytical solution is written as shown in Equation (2.8).

$$u(x;L) = \frac{L^2}{\pi^2}\sin\left(\frac{\pi x}{L}\right) + \frac{2L + 10(L-1)(L-x) - 1}{10\pi} \tag{2.8}$$

The displacement equation of (2.8) is differentiated to calculate the sensitivity of displacement to the length of the bar. The sensitivity equation (2.9) is later used to verify the results of continuum and discrete sensitivity formulations.

$$\frac{\partial u}{\partial L} = \frac{2L}{\pi^2}\sin\left(\frac{\pi x}{L}\right) - \frac{x}{\pi}\cos\left(\frac{\pi x}{L}\right) + \frac{10L - 5x - 4}{5\pi} \tag{2.9}$$

2.3 Discrete Sensitivity Analysis

2.3.1 Formulation

The derivation of the discrete sensitivity equations starts with the discrete form of the governing equations. For the linear case, the governing equation (2.1a) is discretized as shown in Equation (2.10)

$$[K]\,[U] = [F] \tag{2.10}$$

Where $[K]$ is the discrete operator that represents the governing equation; $[U]$ is the vector of response variables defined at each of the degrees of freedom, and $[F]$ is the vector of the applied loads. The sensitivity of this system to design variable b is derived by differentiating Equation (2.10).

$$[K]\left[\frac{\partial U}{\partial b}\right] = \left[\frac{\partial F}{\partial b}\right] - \left[\frac{\partial K}{\partial b}\right][U] \tag{2.11}$$

Solving Equation (2.11) yields a solution for sensitivity of response variable, $\partial u/\partial b$. This is achievable if the values of $\partial F/\partial b$ and $\partial K/\partial b$ are known. It should be noted that U is calculated by solving the governing equation (2.10). For the case of design independent loads, $\partial F/\partial b$ is equal to zero; this happens in many situations such as design optimization with pure mechanical loads. This assumption is violated for the cases when changing the design variables alter the loadings on the structure. The best example for this case is the design of thermally loaded structures where, due to thermal expansions, changing the size of elements will affect the thermal loadings [68]. Calculating the $\partial K/\partial b$ term requires a detailed understanding about the discretization approach. This often requires source code modification that makes this method difficult to implement for commercial codes such as ABAQUS or FLUENT. The discrete method is an accurate approach for calculating the sensitivities as will demonstrated with the following examples; however, it is almost impossible to implement in general-purpose solvers.

2.3.2 Implementation for heat transfer problem

The discrete sensitivity equations are formulated by discretizing the governing equation (2.4) using finite difference method. It should be noted that the finite difference is used for discretization of continuum governing equation and not for sensitivity calculation. Other techniques, such as finite volume or finite elements, can be used for the discretization of the governing equations as well. For this problem, the design variable affects the shape of the domain which is related to the distance between the nodes in a discrete manner. Therefore, for the sake of sensitivity analysis, it is required to keep the nodal distances in the discretized solution as well. The domain is discretized by using six nodes as shown in Figure 2.4.

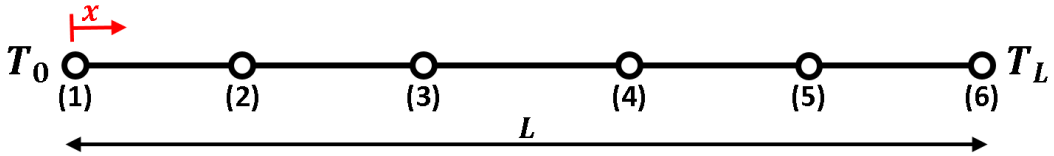


Figure 2.4: One dimensional computational domain for the heat conduction problem.

The second derivative of Equation (2.4) needs to be approximated for discretization. This is done by writing the Taylor series expansion of temperature at arbitrary location x_i. By using the central difference method, the second order approximation for the second order derivative is written as shown in Equation (2.12).

To maintain the generality, we assume that distance of node T_i to T_{i+1} is Δ_i and the distance of node T_i to T_{i-1} is Δ_{i-1}.

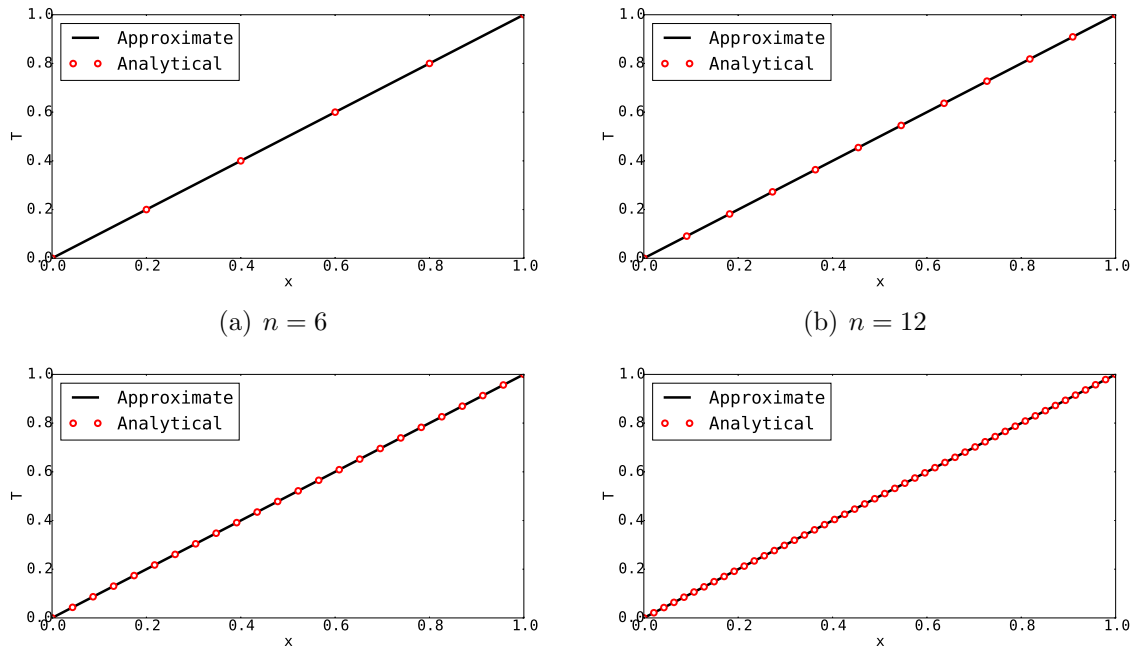
$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1}\Delta_{iL} - T_i(\Delta_{iL} + \Delta_{iR}) + T_{i+1}\Delta_{iR}}{\frac{1}{2}\left[\Delta_{iL}\Delta_{iR}^2 + \Delta_{iL}^2\Delta_{iR}\right]} \tag{2.12}$$

The discretized governing equation of (2.12) is written in a matrix form as shown in Equation (2.13).

$$\begin{bmatrix} \frac{-2}{\Delta_1\Delta_2} & \frac{2}{\Delta_1\Delta_2+\Delta_1^2} & 0 & 0 \\ \frac{2}{\Delta_3^2+\Delta_2\Delta_3} & \frac{-2}{\Delta_2\Delta_3} & \frac{2}{\Delta_2\Delta_3+\Delta_2^2} & 0 \\ 0 & \frac{2}{\Delta_4^2+\Delta_3\Delta_4} & \frac{-2}{\Delta_3\Delta_4} & \frac{2}{\Delta_3\Delta_4+\Delta_3^2} \\ 0 & 0 & \frac{2}{\Delta_5^2+\Delta_4\Delta_5} & \frac{-2}{\Delta_4\Delta_5} \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = - \begin{bmatrix} \frac{2T_1}{\Delta_2^2+\Delta_1\Delta_2} \\ 0 \\ 0 \\ \frac{2T_6}{\Delta_5\Delta_4+\Delta_5^2} \end{bmatrix} \tag{2.13}$$

To verify the discretization process, we compare the analytical solution of this problem with

the result of Equation (2.13) in Figure 2.5. For this problem we choose $T_1 = 0$, $T_6 = 1$ as the boundary conditions and $L = 1$. As shown in this figure, the discrete and continuum results match very well for a different number of nodes. This is done by comparing the absolute error between the analytical and approximate results as shown in Table 2.1.



(a) $n = 6$ (b) $n = 12$

(c) $n = 24$ (d) $n = 48$

Figure 2.5: Comparison between the analytical and finite difference solutions for 1D heat equation for different number of nodes.

Number of nodes	absolute error
6	1.85×10^{-16}
12	2.03×10^{-16}
24	1.27×10^{-15}
48	8.55×10^{-15}

Table 2.1: Absolute error value for different number of nodes.

The discrete sensitivity equations are derived by differentiating the discretized governing equation of (2.13) to the length of the domain. We assume that the change in the length only affects the nodal distance between the last two nodes and the rest remain unchanged. This means that only the node next to the boundary will move, and the rest of nodes will be stationary. This is required to make sure that the sensitivity at each of the degrees of freedom is only a function of the change in shape not the movement of material nodes; therefore, $\partial x/\partial b$ is equal to zero.

In order to differentiate Equation (2.13), it is required to calculate the derivative of nodal distances in Equation (2.13) with respect to L, $\partial\Delta_i/\partial L$. For an equally spaced grid, the nodal distance is written as

$$\Delta = \frac{L}{n-1}$$

Where L is the length of the domain and n is the number of nodes used to discretize the domain. Therefore, the sensitivity of nodal distances to the length of the domain is calculated as:

$$\frac{\partial\Delta}{\partial L} = \frac{1}{n-1} \tag{2.14}$$

The differentiated form of the discretized equation is written as shown in Equation (2.15). It should be noted that since only the adjacent node to the boundary is affected by shape change, only that element in the matrix derivative has a value.

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} T_2' \\ T_3' \\ T_4' \\ T_5' \end{bmatrix} = \frac{1}{2}\frac{\partial\Delta}{\partial L}\frac{1}{\Delta} \begin{bmatrix} 0 \\ 0 \\ 0 \\ T_6 \end{bmatrix} \underbrace{- \frac{1}{2}\frac{\partial\Delta}{\partial L}\frac{1}{\Delta} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\text{effect of shape change}} \tag{2.15}$$

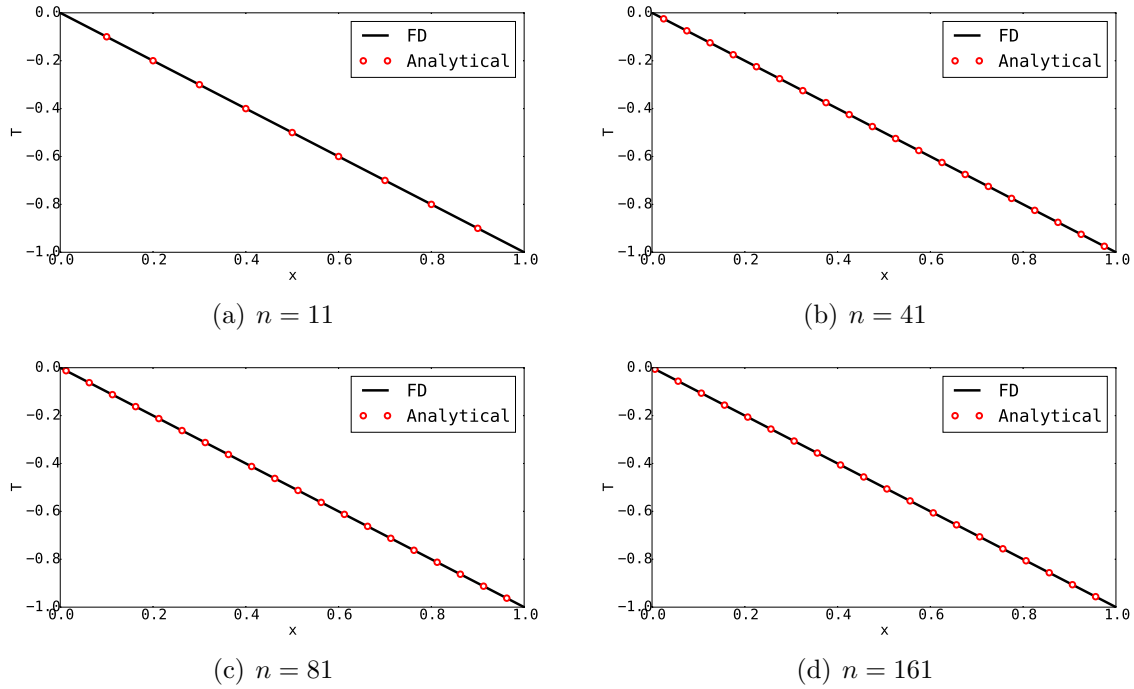
T_i' represents the sensitivity of temperature to length of the domain.

To verify the results of Equation (2.15), we compared it with the analytical sensitivity results as shown in Figure 2.6. We discretized the domain using 11, 41, 81, and 161 nodes for this purpose; however, this does not affect the solution accuracy. We chose the normalized root-mean-square deviation (NRMSD) for comparing the sensitivity results. This is defined in Equation (2.16).

$$NRMSD = \frac{\sqrt{\frac{\sum(\hat{y}_t - y)^2}{n}}}{y_{max} - y_{min}} \tag{2.16}$$

Where $\hat{y}$ is the value calculated using DSA and y is the true value calculated by the analytical

equation. The results for the NRMSD for this problem using different numbers of nodes are shown in Table 2.2.



(a) $n = 11$ (b) $n = 41$

(c) $n = 81$ (d) $n = 161$

Figure 2.6: Comparison between discrete sensitivity analysis and analytical results for different number of nodes.

Number of nodes	NRMSD
11	1.68×10^{-16}
41	2.49×10^{-15}
81	1.49×10^{-15}
161	2.85×10^{-15}

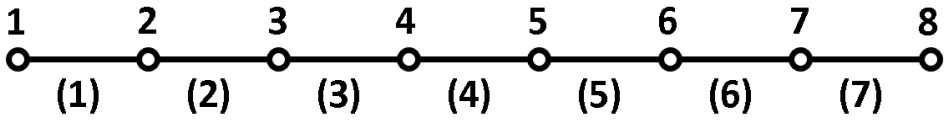
Table 2.2: RMSE value for different number of nodes.

As shown in Figure 2.6 and Table 2.2, the accuracy of discrete sensitivity analysis is not affected by the number of nodes chosen to discretize the domain.

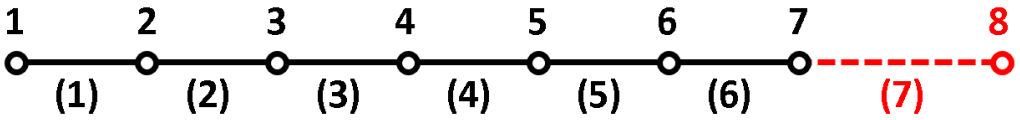
2.3.3 Implementation for solid mechanics problem

Finite elements are used to discretized the axial bar problem is shown in Figure 2.3. The governing equation of (2.7a) is discretized using a uniform mesh, linear shape functions, and one point Gauss integration. As mentioned in Section 2.2.2, the design variable is selected as the length of the bar affecting the distances between the nodes. To keep the local and total derivatives equal, we fix the interior computational nodes in space and change the length of the

bar by only moving the last node. Therefore, the shape design variable only affects the size of the last element and should be included in the discretized equations. This is shown in Figure 2.7. Since the internal nodes do not move, the design velocity is zero for these nodes.



(a) Original domain



(b) Deformed domain

Figure 2.7: Changing the bar length by fixing the interior computational nodes and only moving the boundary (red) node. Node numbers are represented by i and element numbers by (i).

For the case of three elements, the stiffness matrix is written as

$$[K] = EA \begin{bmatrix} 1/l_1 & -1/l_1 & 0 & 0 \\ -1/l_1 & 1/l_1 + 1/l_2 & -1/l_2 & 0 \\ 0 & -1/l_2 & 1/l_2 + 1/l_3 & -1/l_3 \\ 0 & 0 & -1/l_3 & 1/l_3 \end{bmatrix} \tag{2.17}$$

Where l_i are the lengths of the elements. It should be noted that only l_3 is affected by changing the length of the domain in this formulation of the problem. The discretized equation for this problem is written as shown in Equation (2.18).

$$[K - K_s][U] = [F_d] - [F_p] \tag{2.18}$$

Where $[K]$ is the stiffness matrix of the bar; $[K_s]$ is the stiffness of the spring boundary; $[F_d]$ is the distributed load; and $[F_p]$ is the point load. The point load is substracted because of its negative direction for this problem. For the case of $EA = 1$, $F_d = \sin(\pi x/2)$, $L = 1$, $F_p = 1/\pi$, and $K_s = 10$ the solution of Equation (2.18) is compared with the analytical results of the problem as shown in Equation (2.8) in Figure 2.8. We chose Normalized Root-Mean-Square

Error (NRMSE) as the metric of comparison. This is calculated as shown in Equation (2.19)

$$NRMSE = \frac{\sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y)^2}{n}}}{y_{max} - y_{min}} \tag{2.19}$$

where $\hat{y}_t$ is numerical and y is the analytical result.

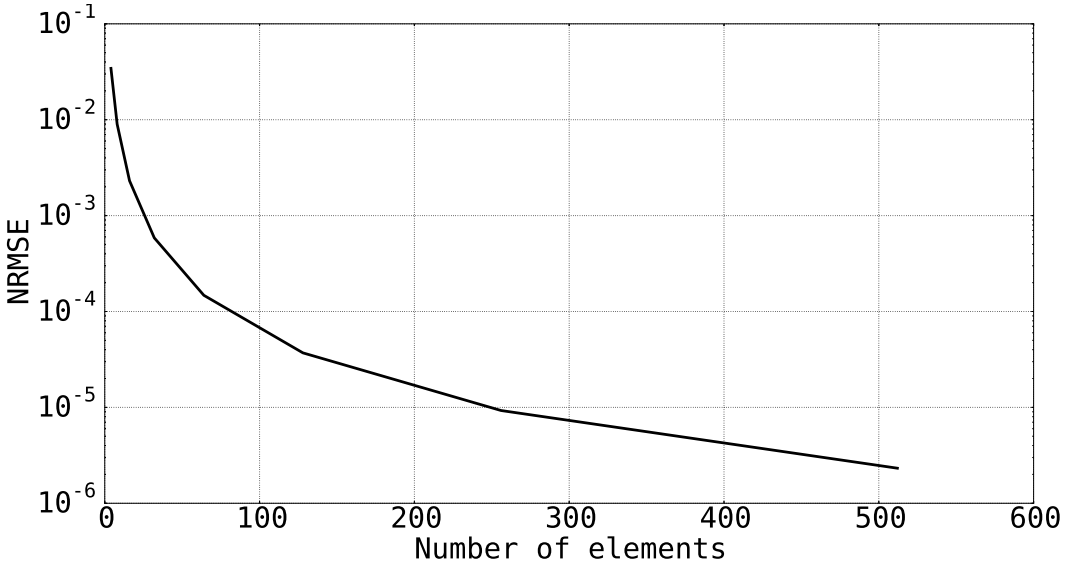


Figure 2.8: Mesh convergence of the finite elements analysis for axial bar.

The discrete sensitivity equation is derived by differentiating Equation (2.18) with respect to the shape of the domain as shown in Equation (2.20).

$$\left[\frac{\partial K}{\partial b} - \frac{\partial K_s}{\partial b}\right][U] + [K - K_s]\left[\frac{\partial U}{\partial b}\right] = \left[\frac{\partial F_d}{\partial b}\right] - \left[\frac{\partial F_p}{\partial b}\right] \tag{2.20}$$

Where b is the design variable. $[K_s]$ is not affected by the design variable; therefore, its derivative with respect to b is equal to zero. The same argument can be made for the point load F_p. Using these observations, Equation (2.20) is simplified as:

$$\left[\frac{\partial U}{\partial b}\right] = [K - K_s]^{-1}\left\{\left[\frac{\partial F_d}{\partial b}\right] - \left[\frac{\partial K}{\partial b}\right][U]\right\} \tag{2.21}$$

In Equation (2.21), $[U]$ is known as the result of solving the governing equation (2.18)that change in and $\frac{\partial F_d}{\partial b}$ is easily calculated since the distributed force is analytically known. $\frac{\partial K}{\partial b}$ is calculated by differentiating Equation (2.17). It should be noted that the change in bar length, L, only affects l_3. Therefore, $\frac{\partial l_3}{\partial L}$ is equal to one. Finally, the sensitivity of the stiffness matrix

to the shape design variable for a simple case of Equation (2.17) is written as:

$$\left[\frac{\partial K}{\partial L}\right] = -EA \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/l_3^2 & -1/l_3^2 \\ 0 & 0 & -1/l_3^2 & 1/l_3^2 \end{bmatrix} \tag{2.22}$$

The sensitivity results of solving the discrete Equation (2.21) are compared with the analytical result from Equation (2.9). This comparison is shown as NRMSE values in Figure 2.9. As shown here, the analytical and discrete sensitivity results agree very well. Moreover, the error decreases as we increase the number of elements.

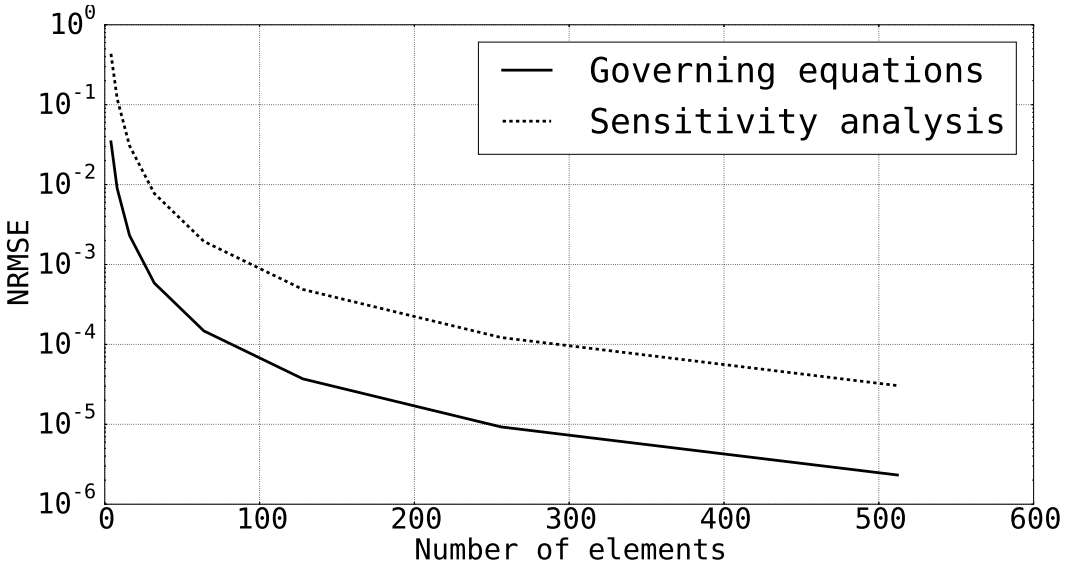


Figure 2.9: Mesh convergence of the discrete sensitivity analysis for the axial bar.

2.4 Continuum Sensitivity Analysis

2.4.1 Formulation

For a continuous system, the governing equations and boundary conditions are written as

$$A(u, t; b) = 0 \qquad \text{on } \Omega \tag{2.23a}$$

$$B(u, t; b) = g(x, t; b) \quad \text{on } \Gamma \tag{2.23b}$$

where u is the response variable, i.e. displacement or pressure, t is time, x is the spatial coordinate, and b is the design variable that can be used to control the solution. A and B are continuum functions that define the governing equations and boundary conditions respectively. It should be noted that the governing equation is written in the residual form where its value needs to be equal to zero when u is the solution at time t. g is the value of the boundary condition of the defined system. To calculate the response sensitivity, the total derivative of Equation (2.23) is required. This is done by determining the local sensitivity of the governing equation followed by converting the local sensitivities to their total form using Equation (2.2).

The governing equation while the boundary condition of (2.23) is differentiated as follows

$$A(u', t; b) + A'(u, t; b) = 0 \qquad \text{on } \Omega \tag{2.24a}$$

$$B(\dot{u}, t; b) + \dot{B}(u, t; b) = \dot{g}(x, t; b) \quad \text{on } \Gamma \tag{2.24b}$$

where $(\ )'$ and $\dot{(\ )}$ are local and total derivative which are defined as follows

$$(\ )' = \frac{\partial(\ )}{\partial b}$$
$$\dot{(\ )} = \frac{D(\ )}{Db}$$

It should be noted that the governing equations are differentiated in the local form, and the boundary conditions are differentiated in the total form. Local differentiation of the governing equations enables us to treat the solver non-intrusively [22]. However, to capture the effect of shape change, the boundary conditions need to be differentiated in the total form. The boundary condition definition is further simplified by assuming linearly. This is a valid assumption for many practical cases such as structural analysis or computational fluid dynamics. The boundary conditions are typically in the form of known gradient or values, i.e. outflow and free-slip wall for CFD and predefined displacement/force for structural analysis. It should be noted that this assumption is not valid for problems such as contacts. The boundary condition is written as:

$$
\begin{aligned}
&B(\dot{u}, t; b) = \dot{g}(x, t; b) \Rightarrow \\
&B(u', t; b) = \dot{g}(x, t; b) - B(\frac{\partial u}{\partial x} \cdot \frac{\partial x}{\partial b}, t; b)
\end{aligned}
\tag{2.26}
$$

To study the differentiated governing equation of Equation (2.24), we will assume two situations: i) linear analysis and ii) nonlinear analysis.

For the linear analysis, the governing of (2.24) is written as:

$$
A(u', t; b) = 0 \tag{2.27}
$$

This is valid for the linear case because the differential operators of the governing equation do not depend on the response variable, u. To explain this concept we look at the transient heat conduction in the 1D domain. The governing equations are defined as

$$
\frac{\partial^2 T}{\partial x^2} = \frac{1}{\alpha} \frac{\partial T}{\partial t} \tag{2.28}
$$

where T is the temperature, x is spatial coordinate, t is time, and α is is the thermal diffusivity ($k/\rho c_p$). This equation can be differentiated with respect to a design variable b as shown in the following equation.

$$
\frac{\partial}{\partial b} \left(\frac{\partial^2 T}{\partial x^2} \right) = \frac{\partial}{\partial b} \left(\frac{1}{\alpha} \frac{\partial T}{\partial t} \right)
$$

Due to linearity of the differential operators we can change the order of differentiation as follows:

$$
\frac{\partial^2}{\partial x^2} \left(\frac{\partial T}{\partial b} \right) = \frac{1}{\alpha} \frac{\partial}{\partial t} \left(\frac{\partial T}{\partial b} \right) \tag{2.29}
$$

By comparing Equations (2.28) and (2.29) we can see that the differential operators $\partial^2/\partial x^2$ and $\partial/\partial t$ remain unchanged. Therefore the same solver can be used for solving this system of governing equations and for the sensitivity variable, $\partial T/\partial b$.

For nonlinear problems, the differential operators are functions of response variables as well. For example, the incompressible Euler's equation for a 2D flow is derived as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} = 0 \tag{2.30a}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} = 0 \tag{2.30b}$$

where u and v are the velocity components in x and y directions respectively. p is pressure, t is time, and x and y are spatial coordinates. We can rewrite Equation (2.30) in terms of differential operators as well.

$$\mathcal{T}u + \mathcal{C}u + \mathcal{G}_x p = 0$$

$$\mathcal{T}v + \mathcal{C}v + \mathcal{G}_y p = 0$$

where $\mathcal{T}$ is the time derivative operator ($\partial/\partial t$) and $\mathcal{C}$ is the convective operator as shown in Equation (2.32).

$$\mathcal{C} = u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} \tag{2.32}$$

$\mathcal{G}_x$ and $\mathcal{G}_y$ are gradient operators in x and y respectively ($\partial/\partial x$ and $\partial/\partial y$). The gradient and time derivative operators are linear; therefore, they can be treated as shown in the previous paragraphs. On the other hand, the convective operator is nonlinear and should be differentiated with response variables as well. As a result, the sensitivity equations can be written in the operator form as

$$\mathcal{T}u' + \mathcal{C}'u + \mathcal{C}u' + \mathcal{G}_x p' = 0 \tag{2.33a}$$

$$\mathcal{T}v' + \mathcal{C}'v + \mathcal{C}v' + \mathcal{G}_y p' = 0 \tag{2.33b}$$

where

$$\mathcal{C}' = u'\frac{\partial}{\partial x} + v'\frac{\partial}{\partial y}$$

Several interesting properties of CSA can be explained using Equation (2.33). Importantly, although the original Euler equation is nonlinear due to the multiplication of response variables and their derivatives (u and $\partial u/\partial x$), the resulting sensitivity equation is linear. This means that the sensitivity equations are easier to solve both concerning algorithms and simulation time compared to the original equations. The challenging expression that needs to be calculated in Equation (2.33) is the convective term derivative.

The first step in solving the sensitivity equations is to find the solution of the governing equations. This enables us to calculate the convective operator, $\mathcal{C}$, at each step of sensitivity solution based on the analysis data. $\mathcal{C}'$ is also calculated at each step of the solution of sensitivity equations based on the solution of the previous time step. For a simple predictorcorrector method used to solving the original governing equations, this can be written as follows for sensitivity equation in x direction.

$$\bar{u}' = u'(i) - \Delta t\left[\mathcal{C}'(i)u(i) + \mathcal{C}(i)u'(i) + \mathcal{G}p'(i)\right] \quad : \text{predictor}$$

$$u'(i+1) = u'(i) + \frac{\Delta t}{2} - \left[\mathcal{C}'(i)u(i) + \mathcal{C}(i)u'(i) + \mathcal{G}p'(i) + \bar{\mathcal{C}}(i)u(i) + \mathcal{C}(i)\bar{u} + \mathcal{G}p'(i)\right] \quad : \text{corrector}$$

In the above equations, $\bar{\mathcal{C}}$ in the convective operator is evaluated using $\bar{u}$. It should be noted that to generate the operator we do not need to know the details of how it has been put together. We only supply the required material for generating the convective operator; the black-box solver will generate it for us. This input is the response variable when solving the governing equation and is the sensitivity of response variable for sensitivity analysis. Therefore, the solver can still be considered as a black box that we do not modify.

2.4.2 Implementation for heat transfer problem

In this section, the continuum sensitivity formulation is used for calculating the sensitivity of temperature with respect to the length of a 1D domain. The domain is defined in Figure 2.2 with the temperature in the domain governed by Equation (2.4). The boundary conditions are defined as T_0 at $x = 0$ and T_L at $x = L$.

To get the sensitivity equations, governing equations are differentiated in the local form and

boundary conditions are differentiated in total form as shown in Equation (2.34).

$$\frac{\partial}{\partial L}\left(\frac{\partial^2 T}{\partial x^2} = 0\right) \tag{2.34}$$

Since the differential operator is linear, the order of differentiations is changed. This will give us the Equation (2.35) for the sensitivity calculation.

$$\frac{\partial^2}{\partial x^2}\left(\frac{\partial T}{\partial L}\right) = 0 \tag{2.35}$$

The boundary conditions for this problem are written as follows to be consistent with the general formulation of the boundary conditions.

$$\begin{cases} \mathcal{B}T = T_0 & \text{at x = 0} \\ \mathcal{B}T = T_L & \text{at x = L} \end{cases} \tag{2.36}$$

Where $\mathcal{B}$ is the operator acting on the boundary. For this problem, it is equal to 1. Equation (2.36) is differentiated in the total form since the boundaries are moving. This results in

$$\begin{cases} \dot{\mathcal{B}}T + \mathcal{B}\dot{T} = \dot{T}_0 & \text{at x = 0} \\ \dot{\mathcal{B}}T + \mathcal{B}\dot{T} = \dot{T}_L & \text{at x = L} \end{cases} \tag{2.37}$$

where $\mathcal{B}$ and $\dot{T}_0$ are constants and have zero derivatives. $\dot{T}$ is written in terms of the local derivative and convective terms using the chain rule. This results in the following definition for the sensitivity equation boundary conditions.

$$\begin{cases} \dfrac{\partial T}{\partial b} = -\dfrac{\partial T}{\partial x}\dfrac{\partial x}{\partial b} & \text{at x = 0} \\ \dfrac{\partial T}{\partial b} = -\dfrac{\partial T}{\partial x}\dfrac{\partial x}{\partial b} & \text{at x = L} \end{cases}$$

To calculate the mesh sensitivity at the boundary, the dependency of the spatial variable, x, and the shape of the domain, L, need to be known. For the 1D domain, this is defined as follows:

$$x = \alpha L$$

This means that every location in the domain can be defined using a nondimensional variable, α, times the total length of the domain. Using this formulation, the sensitivity of spatial coordinate, x, with respect to the length of the domain is defined as:

$$\frac{\partial x}{\partial L} = \alpha \quad \text{where} \quad \alpha = \frac{x}{L}$$

By using this relation, the boundary conditions can be written as:

$$\begin{cases} \mathcal{B}\dfrac{\partial T}{\partial b} = -\dfrac{\partial T}{\partial x}\dfrac{x}{L} & \text{at x = 0} \\ \mathcal{B}\dfrac{\partial T}{\partial b} = -\dfrac{\partial T}{\partial x}\dfrac{x}{L} & \text{at x = L} \end{cases}$$

This can be further simplified by substituting x in the definition of boundary conditions. We also substitute $\mathcal{B}$ as 1.

$$\begin{cases} \dfrac{\partial T}{\partial b} = 0 & \text{at x = 0} \\ \dfrac{\partial T}{\partial b} = -\dfrac{\partial T}{\partial x} & \text{at x = L} \end{cases} \tag{2.38}$$

The last thing to be noted is the spatial derivative of response variable at the boundaries, $\partial T/\partial x$. This term appears due to the boundary movement with a change in shape design variable when using the chain rule. This term is calculated from the analysis results by using the same technique used for discretizing the governing equations. For this problem, the finite difference method is used to calculate this derivative from the analysis results.

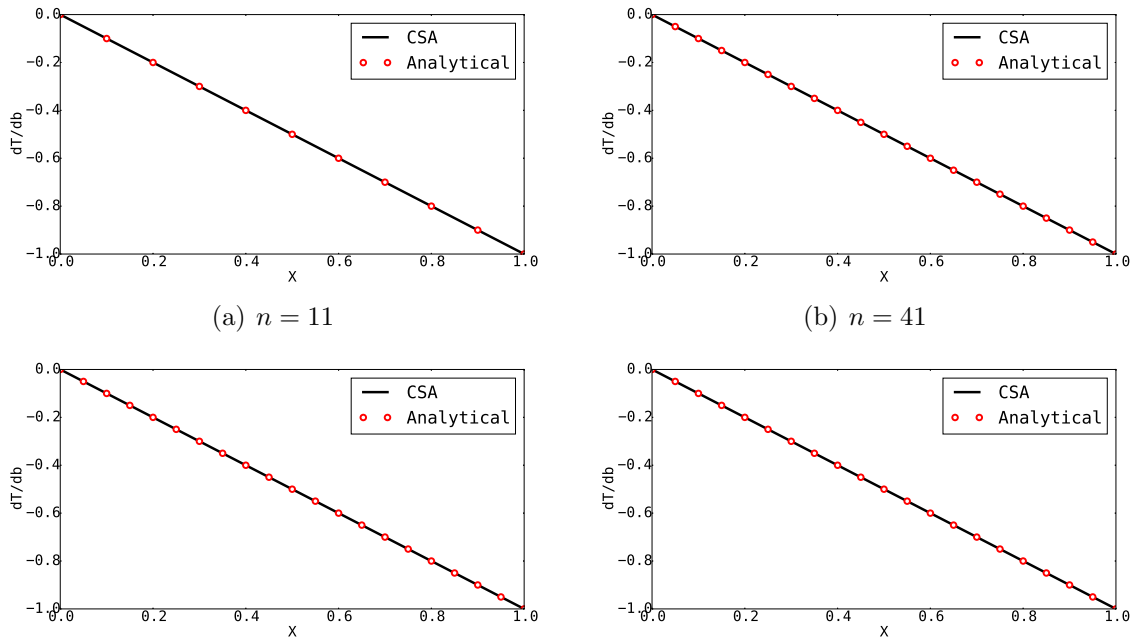
The governing equation of (2.35) with boundary condition (2.38) is solved using the same solver that we used for solving the original governing equation since these two are effectively having the same form. The comparison between the original governing equation and the sensitivity equation is shown in Table 2.3.

Analysis Type	Equation	Unknowns	Discretized form	Discretization method
Governing equation	$\partial^2 T/\partial x^2$	T	$[K][T] = [F_{BC}]$	Central difference
Sensitivity equation	$\partial^2 T'/\partial x^2$	T'	$[K][T'] = [F'_{BC}]$	Central difference

Table 2.3: Comparison between the governing and sensitivity equations.

The continuum sensitivity analysis (CSA) result is verified using the analytical solution of

Equation (2.5) as shown in Figure 2.10. Different number of discretization nodes are used for the comparison between the results. For the qualitative comparison between the results, the NRMSD as defined in Equation (2.16) is used. As shown if Figure 2.10 and Table 2.4, the two results match perfectly.



(a) $n = 11$ (b) $n = 41$

(c) $n = 81$ (d) $n = 161$

Figure 2.10: Comparison between continuum sensitivity analysis and analytical results for different number of nodes.

Number of nodes	NRMSD
11	6.96×10^{-17}
41	2.02×10^{-15}
81	1.43×10^{-15}
161	2.77×10^{-15}

Table 2.4: RMSE value for different number of nodes.

It should be noted that the sensitivity equations are derived and solved in the local form. This does not include the effect of nodes moving in the computational domain. If the analysis requires the sensitivity at the material points, an additional step is required to convert the local sensitivities to their total form using the chain rule as shown in the following equation.

$$\frac{DT}{Db} = \frac{\partial T}{\partial b} + \frac{\partial T}{\partial x} \cdot \frac{\partial x}{\partial b}$$

This will add an extra cost to the simulation due to the additional step for calculating

$\partial x/\partial b$. The reason this step exists is due to movement of computational nodes as the results of a change in shape. Therefore, if the computational nodes can be fixed, the $\partial x/\partial b$ term will be equal to zero. We are proposing to achieve this using a non-body conformal technique such as immersed boundary method. This will be explained in the next chapter in more detail.

2.4.3 Implementation for solid mechanics problem

To derive the continuum sensitivity equations, the governing equation of (2.7a) is differentiated with respect to the design variable, L, as shown in Equation (2.39).

$$\frac{\partial^2}{\partial x^2}\left(\frac{\partial u}{\partial L}\right) - \frac{\pi x}{L^2}\cos\left(\frac{\pi x}{L}\right) = 0 \tag{2.39}$$

The boundary conditions are calculated by differentiating the boundary conditions defined in Equation (2.7b). It should be noted that the boundary conditions are differentiated in total form.

$$\begin{cases} \dfrac{\partial}{\partial x}\left(\dfrac{\partial u}{\partial L}\right)\Big|_{x=0} = \dfrac{\partial}{\partial L}\left(\dfrac{1}{\pi}\right) = 0 \\ \dfrac{\partial}{\partial x}\left(\dfrac{\partial u}{\partial L}\right)\Big|_{x=L} = \left\{K\left[\dfrac{\partial u}{\partial L} + \dfrac{\partial u}{\partial x}\dfrac{\partial x}{\partial L}\right] - \dfrac{\partial^2 u}{\partial x^2}\dfrac{\partial x}{\partial L}\right\}\Big|_{x=L} \end{cases} \tag{2.40}$$

For this problem, the spatial coordinate x is defined as $x = \zeta L$ where $\zeta = x/L$. Therefore, the design velocity $\partial x/\partial L$ at $x = L$ is equal to one. Solving the governing equation (2.39) along with boundary conditions (2.40) leads to the sensitivity values of displacement in the domain, $\partial u/\partial L$. Comparing the sensitivity equation and boundary conditions (2.39) and (2.40) to the governing equation and its corresponding boundary conditions of (2.7a) and (2.7b), it is safe to say that these equations are the same. The only difference is their boundary conditions. Therefore, the same numerical solver that was used for solving the governing equation can be used to solve the sensitivity equations without any modifications. The only change would be in the values of the boundary conditions. However, change in the boundary conditions does not alter the structure of the black box solver.

To calculate the sensitivities, the governing equation and boundary conditions of (2.39) and (2.40) are discretized using the finite element method as discussed through equations (2.17) and (2.18). It should be noted that $[K]$ and $[K_s]$ are the same matrices that used in Equation (2.17),

$[F_p]$ is equal to zero, and $F_d = -\frac{\pi x}{L^2}\cos\left(\frac{\pi x}{L}\right)$. For verification, the results for the sensitivity analysis are compared with the analytical results of sensitivity as defined by Equation (2.9). We used the NRMSE metric to compare the two different results as defined in Equation (2.19). The comparison of the results are shown in Figure 2.11. As shown here, the results for both discrete and continuum sensitivity analysis converge with the same rate.

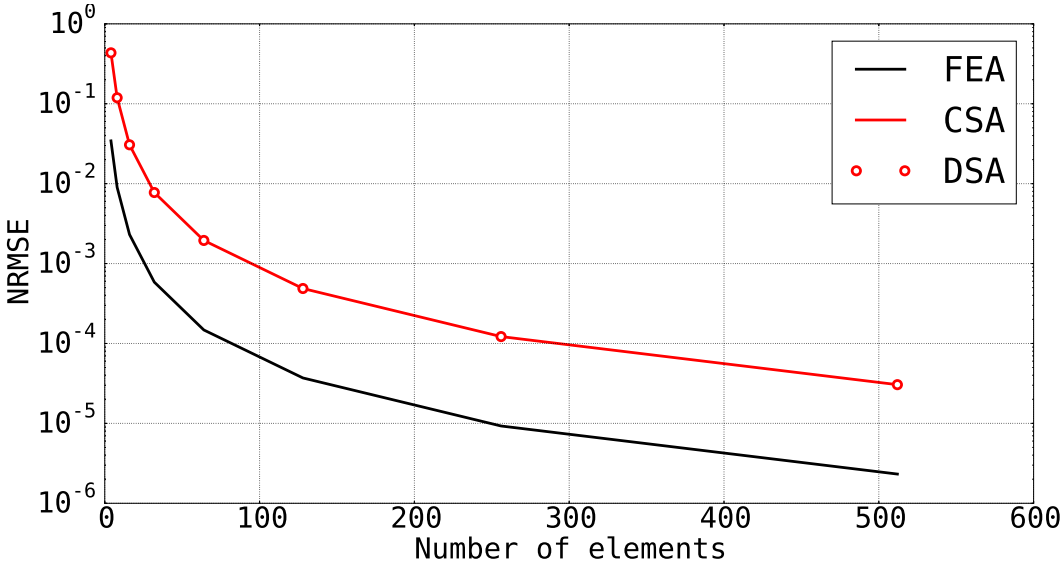


Figure 2.11: Mesh convergence of the continuum (CSA) and discrete (DSA) sensitivity equations for axial bar problem. The results for the convergence of the analysis (FEA) is also included in this graph.

2.5 Summary

In summary, we looked at two main approaches used to calculate the sensitivity response of a system. The discrete method is based on differentiating the discretized governing equations whereas in continuum method, the governing equation are differentiated first and then discretized. We applied these techniques to a simple 1-D heat transfer problem and calculated the sensitivity of response to shape design parameter. The sensitivities are calculated in the local form for each of these methods and compared with the analytical results where they show good comparison. It is shown that by using the continuum sensitivity analysis, the differential operators used in the solution of governing equations are reused in the sensitivity analysis. This effectively means that the black-box solver used in the simulation step can be reused with new boundary conditions for solving the sensitivity response. As shown in the example problem, this is not possible when using discrete method. This is mainly due to the fact that the discrete

operators will be affected by differentiating the discretized governing equations with respect to design variables. Therefore, more details about the analysis needs to be known when using discrete approach compared to the continuum. Continuum sensitivity approach is used in this research due to its ability to treat the analysis as black-box for solving both the governing equations and the sensitivity response. However, the result of sensitivity analysis is still the local sensitivities which needs to be transformed to the total form using the chain rule. This is not favorable because it is another step on top of an already expensive analysis. We are proposing to use a non-body conformal approach such as immersed boundary method for analysis to remove this step from the sensitivity calculation.

Chapter 3

Immersed Boundary Method

Immersed boundary methods are a class of techniques in computational fluid dynamics where the governing equations are solved on a Cartesian grid that does not conform to the shape of the body in the flow. This treatment of the solid boundaries is opposed to well known body-conformal techniques where the computational mesh accurately represents the shape of the domain. The boundary condition on the immersed surfaces are not applied explicitly, instead, an extra forcing function is added to the governing equations, or the discrete numerical scheme is updated near the boundary. The immersed boundary technique is, in particular, interest to us since it removes the mesh sensitivity calculation from the analysis. In this chapter, we talk about different classes of immersed boundary technique and apply them to a simple problem. The applicability of these methods in the continuum sensitivity analysis is also discussed. At the end of this chapter, we will choose a couple of immersed boundary techniques for sensitivity analysis implementation.

3.1 Introduction

When people started to use computational models in the design of systems, it was usually sufficient to include single physics into the design. The simulations were usually based on structural solvers using finite elements analysis (FEA) or computational fluid dynamics (CFD) simulations. The new requirements such as higher fuel efficiency, improved controllability, higher stiffness to mass ratios and lower radar signature have forced the designers to develop more unconventional configurations. For example, one way to reduce the infra-red signature

of the aircraft, is to remove the engines from underneath the wings and put them inside the aircraft. However, by doing this a massive heat source will be added to the structure. The thermal expansion due to this excessive heat load needs to be incorporated into the structural analysis of the system. This requires a multidisciplinary analysis combining thermal analysis for heat transfer and structural analysis for thermal expansion and other structural loads [68]. However, this has not been a traditional design method. Only recently people have started to look into such configurations.

The multidisciplinary analysis is required for many engineering applications, however, the one that is used the most is the interaction of fluid and a deformable structure. This problem is commonly known as a Fluid-Solid Interaction (FSI) problem. FSI problems are dealt with in many different engineering applications, such as fluttering and buffeting of bridges [69], the vibration of wind turbine blades [70], and aeroelastic response of airplanes [71]. FSI problems are also seen in blood flows in arteries and artificial heart valves [72], flying and swimming [73]. The conventional approach for simulating such problems is the Arbitrary Lagrangian-Eulerian (ALE) method. ALE methods are based on the body-conforming grids to track the location of the fluid-structure interface. ALE methods have been applied to many FSI problems. However, they are cumbersome if not impossible to apply to FSI problems with large deformations for complicated boundary shapes.

Immersed boundary (IB) methods are considered a separate family of methods used for modelling FSI problems with complex boundary shapes and large deformations. IB methods are based on solving the governing equations for fluids on a fixed grid. Although this computational grid can be structured (Cartesian) or unstructured, most methods are based on structured grid because of the simplicity of grid generation. Moreover, when using structured grid, extremely efficient computational methods can be utilized on to solve the governing equations. In IB approach, the fluid-structure boundaries are represented by a set of independent nodes. The solid boundary effect on the flow is formulated either by introducing fictitious body forces in the governing equations or by locally modifying the structure of the background grid.

IB has several advantages over the ALE technique. Probably the biggest advantage is the simplification of the task of grid generation. Generating body-conformal grid for a complex shape is usually very complicated. The objective is to construct a grid that provides adequate

local resolution with the minimum number of total grid points. The grid generation requires a significant input from the user and is an iterative process. For complicated boundaries, the unstructured grid approach is better suited however the grid quality is reduced for extremely complex geometry. In contrast, for a simulation carried using an IB method, grid complexity and quality are not affected by the complexity of the geometry.

This advantage becomes, even more, evident for flows with moving boundaries. The ALE approach requires generating a new mesh or deforming the original mesh to match the new boundary shape at each time step. The solution from last time step is also required to be projected to this new computational mesh. Both of the deformation/projection can affect the accuracy, robustness and the computational cost associated with the simulation. On the other hand, the boundary motion in IB method can be handled with rather ease because the computational mesh does not depend on the shape of the boundary. Therefore, although a significant progress in simulating flows using ALE methods has been made in the recent years [74, 75, 76], the IB method remains an attractive alternative for such problems due to its simplicity and cost.

In the following sections of this chapter we first look at the governing equations for the fluid and solid domain. Followed by this different approaches for modelling the follow using IB method are discussed in detail. We apply different IB techniques to a simplified problem to compare the efficiency and simplicity of implementation. The results are compared with body-conformal solution approach for accuracy comparison. Finally, we assess different IB methods with regards to their applicability of the Continuum Sensitivity Analysis (CSA) framework.

3.2 Governing Equations

In the fluid region Ω_f, the governing equations for incompressible flow of a Newtonian fluid are known as Navier-Stokes (NS) equations. In the compact indicial notation, the NS equations are written as

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.1a}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{1}{\rho_f}\frac{\partial p}{\partial x_i} + \nu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + f_i \tag{3.1b}$$

where the repeated indices imply summation and $i, j = 1, 2, 3$. x_i are the spatial coordinates, u_i are the velocity components of the fluid in i direction, ρ_f is the fluids density, p is the pressure, ν is the kinematic viscosity, and f_i are body forces. These forces are used in the IB technique to represent the effect of immersed boundaries on the fluid. In general purpose CFD solvers, due to the use of an unstructured grid to represent the shape, it is usually required to use mappings (Jacobian) to convert the physical coordinate to a computational coordinate. This will become problematic when we have skewed elements that cause the mapping to become singular. This step is removed in the IB approach since the governing equations (3.1) are discretized on a Cartesian grid.

The solid domain is modelled using a linear elastic theory in this work. The governing equations are written as

$$\sigma_{ji,j} + F_i = \rho_s \partial_{tt} u_i \tag{3.2a}$$

$$\epsilon_{ij} = \frac{1}{2}\left(u_{j,i} + u_{i,j}\right) \tag{3.2b}$$

$$\sigma_{ij} = C_{ijkl}\epsilon_{kl} \tag{3.2c}$$

Where σ is the Cauchy stress tensor, ϵ is the strain tensor, u is the displacement vector, C is stiffness tensor, F is the body force, and ρ_s is the mass density of solid. These governing equations are solved to track the motion of the solid boundary.

To solve the coupled system of equations of (3.1) and (3.2), we need to have boundary conditions. The boundary conditions of (3.1) are defined as pressure or velocity magnitudes on the outer boundaries of the domain. After solving Equation (3.1), the loads on the structure, are calculated by integrating the fluid pressure from over the solid boundaries. The resulting force will be the boundary load used for solving Equation (3.2). When the governing equations for the solid region are solved, the displacements of the solid domain are known. This is fed into the CFD solver to update the solid boundary location for the fluid domain. This will

change the solution of fluid's solver resulting in different pressure distributions. This process is repeated until convergence is satisfied. The convergence can be defined as a change in the structural deflection for two subsequent steps in a static problem.

3.3 Benchmark Case

We define a 1D benchmark problem to investigate different IB formulations in detail. In one dimensional space the equations of incompressible flow are not challenging and also it is not simple to write a one-dimensional analogue of the IB to capture all of its features. However, for a simpler one dimensional problem it is easier to understand different formulations of IB method and later apply IB to higher dimensions. Moreover, the sensitivity analysis is better understood. The other reason for working with a simplified model is the availability of analytical results that can be used for verifying the results we get from the numerical solvers. As the final note, for this benchmark case we mainly focused on the fluid domain, the structural behaviour can be easily added to this formulation.

To derive the one dimensional benchmark case we start with the Navier-Stokes equations. Consider a viscous incompressible fluid in the channel $0 \geq y \leq 1$, $-\infty < x < \infty$ as shown in Figure 3.1.

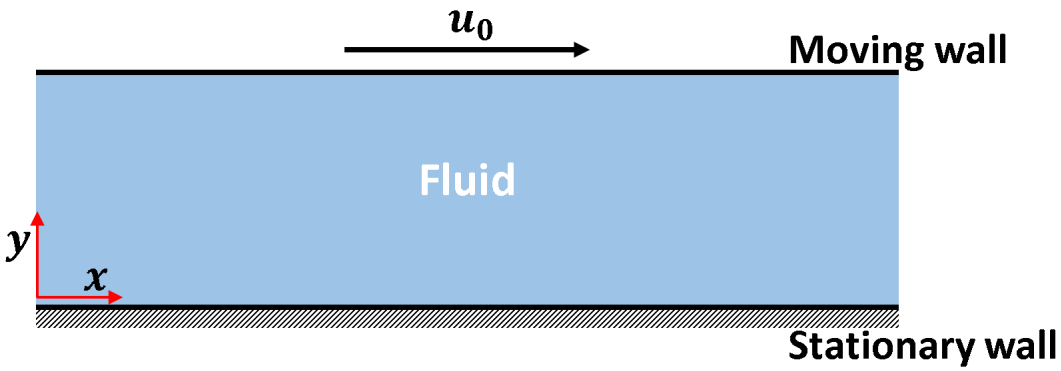


Figure 3.1: 1D benchmark case for IB method.

We can also assume that the boundary conditions are periodic in x. Assume that there is a horizontal plate running through the length of this channel at $y = y_0$ and moving with u_0 velocity in the horizontal direction. We assume no-slip condition where the fluid and the plate meet. The wall velocity will force the fluid to accelerate in x direction due to the viscous stress from the moving plate. We expect no motion in y direction and drops all terms in the NS

equations containing u_j velocity. Due to the continuity equation (3.1a), there is no variation in x direction so we can drop all the terms that involve with x variation as well. This enables us to simplify the NS equation of (3.1b) to Equation (3.3). It should be noted that the continuum equation (3.1a) is automatically satisfied.

$$u_t = \mu u_{yy} \quad \text{in } \Omega_f \tag{3.3a}$$

$$\begin{cases} u = u_0 & \text{at } y = 1 \\ u = 0 & \text{at } y = 0 \end{cases} \tag{3.3b}$$

Equation (3.3) is transient in nature however, its steady state solution can be calculated by setting the time derivative in Equation (3.3) equal to zero. This equation is commonly known as Couette flow in fluid dynamics. The analytical solution for this equation is shown in Equation (3.4).

$$u = u_0 x \tag{3.4}$$

We will use this analytical solution to verify the results of different IB methods defined in the following sections.

3.4 Immersed Boundary Classification

In general, IB methods can be classified in three main families: i) discrete forcing, ii) continuum forcing, and iii) cut-cell methods. This classification is based on how the interface conditions are handled in the IB algorithm. In this section we present the essence of each method and try to point out their primary advantages and disadvantages. In general, the immersed boundary approach is based on modifying the NS equations for imposing the boundary conditions and leads to one of the three categories.

3.5 Discrete Forcing Method

The discrete forcing approach was first introduced in the works of Mohd-Yusof [47] in the late nineties. The discrete forcing approach is based on defining the force term in the numerical solution from the known characteristic of the response. For the case of flow over immersed boundaries, this known feature is the flow velocity on the immersed boundary. The general formulation for the discrete forcing method starts with the discretized NS equation

$$\frac{u^{n+1} - u^n}{\Delta t} = RHS^{n+1/2} + f^{n+1/2} \tag{3.5}$$

where u^{n+1} is the velocity at the next time step, u^n is the velocity at the current time step, Δt is the time step, and $RHS^{n+1/2}$ contains both the viscous terms and pressure gradient. $f^{n+1/2}$ is the forcing term that needs to be calculated in such a way that yields $u^{n+1} = V$, where V is the known velocity of the immersed boundary. Therefore, the force term is calculated as

$$f^{n+1/2} = -RHS^{n+1/2} + \frac{V - u^n}{\Delta t} \tag{3.6}$$

This is valid only if the location of the immersed boundary coincides with the computational nodes. Interpolation from the fluid's computational nodes to the immersed boundary is needed for the cases where the two do not coincide. Depending on how this interpolation is done and where the force terms are calculated, the discrete forcing immersed boundary method is put into two main family based on this.

3.5.1 Indirect forcing method

In the indirect forcing method, the boundary conditions are imposed through a set of forcing terms on nodes near the boundary. These forcing terms are calculated based on the interpolation of a known condition on the immersed boundary from the discretized equations. The forces are applied using delta functions to nodes near the immersed boundary. The interpolation is done as shown in Figure 3.2.

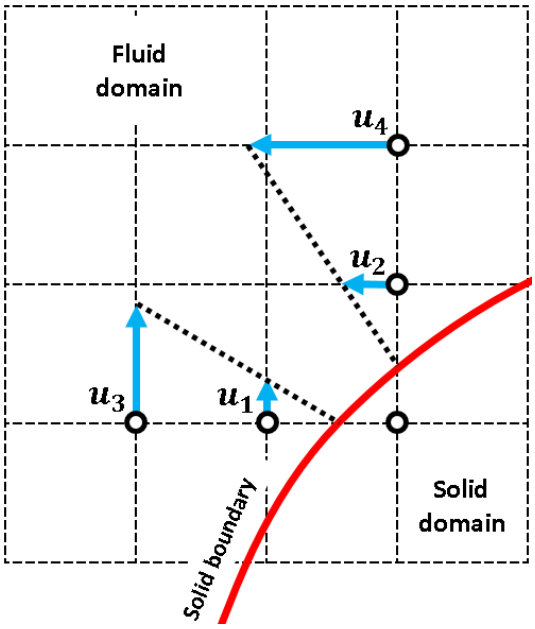


Figure 3.2: Indirect forcing approach for boundary representation. Desired velocity values at nodes 1 and 2 are interpolated from wall velocity and results from nodes 3 and 4.

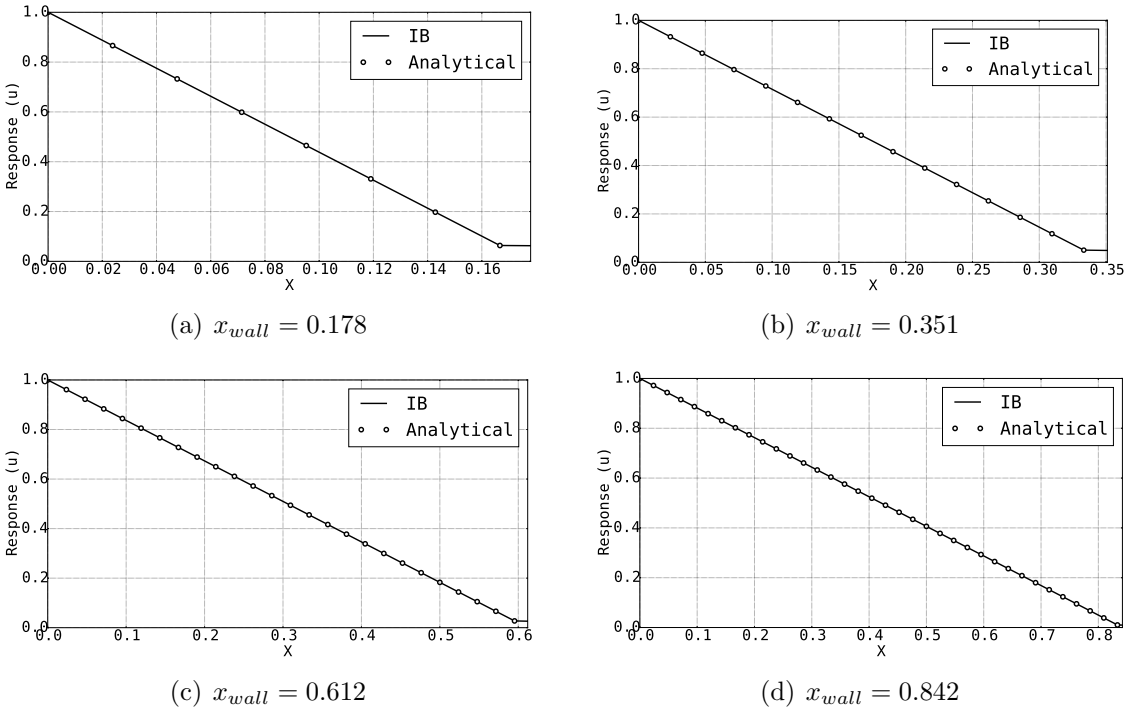
The force terms are applied on nodes 1 and 2 based on the known zero velocity on the wall, and calculated velocities at nodes 3 and 4. Linear interpolation is used to calculate required velocities at nodes 1 and 2 so that it results in zero velocity on the immersed boundary. These known velocities are then employed in Equation 3.6 to calculate the forcing values. This forcing value is then used in Equation 3.5 to compute the velocity field at the following time step.

To investigate the indirect forcing approach, we apply this methodology to the benchmark problem. We investigated three different cases: effect of wall location, the impact of the number of nodes, and effect of wall velocity. In the first case, the wall velocity was $1m/s$ with 41 nodes to discretize the domain. The time step of this simulation is selected as 10^{-3}. The governing equations are discretized using the Euler method in time and a second-order finite difference method in space as shown in the following equation.

$$\frac{u(i,n+1)-u(i,n)}{\Delta t} = \frac{u(i-1,n)-2u(i,n)+u(i+1,n)}{(\Delta x)^2} + f(i,n) \tag{3.7}$$

where $u(i,n)$ is the fluid velocity at location i and time n, Δx is the nodal distances in space, Δt is the nodal distance in time, and $f(i,n)$ is the forcing function value at location i in time n. The forcing function is calculated using the linear interpolation based on zero slip condition on the wall. The results for different locations of the fixed wall are shown in Figure 3.3 and Table 3.1. The wall location is selected as 0.178, 0.351, 0.612, 0.842. As shown here,

the wall location with respect to computational nodes does not affect the solution accuracy.



(a) $x_{wall} = 0.178$ (b) $x_{wall} = 0.351$

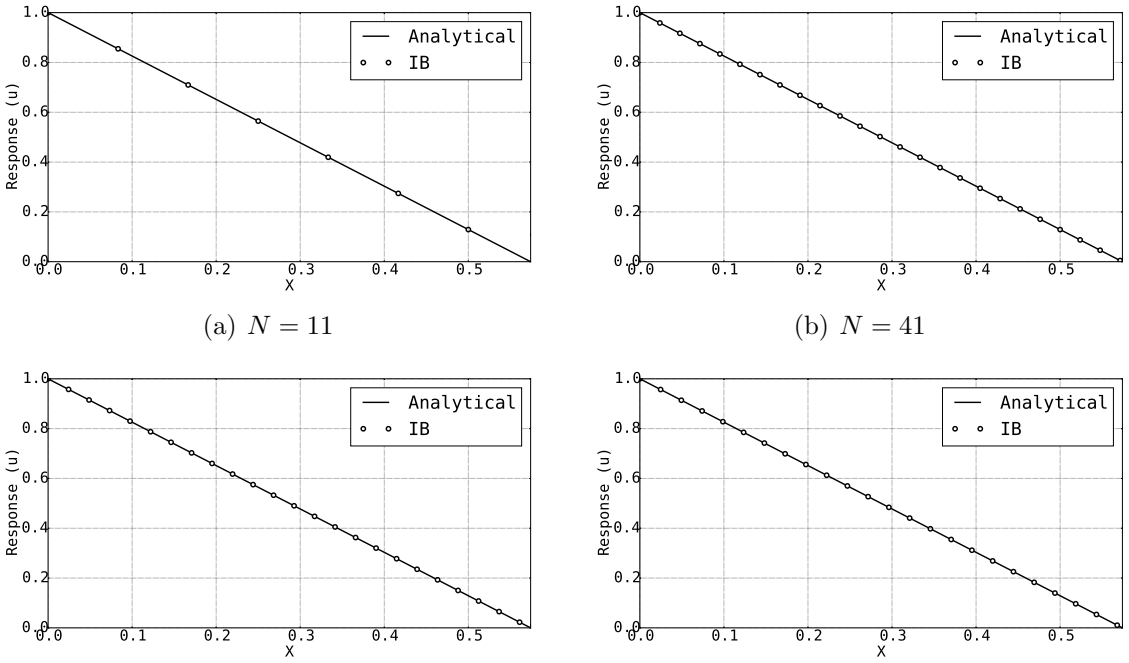
(c) $x_{wall} = 0.612$ (d) $x_{wall} = 0.842$

Figure 3.3: Comparison between IB and analytical results for different wall locations.

Wall location	RMSE value
0.178	1.61×10^{-15}
0.351	2.29×10^{-15}
0.612	1.72×10^{-15}
0.842	3.81×10^{-15}

Table 3.1: RMSE value for different wall locations.

For the second case, we looked at the effect of the number of nodes on the accuracy of the simulation using indirect forcing method. The wall velocity was fixed at $1m/s$ and the stationary wall is located at 0.5741. The number of nodes is chosen as 11, 41, 81, and 161. As can be seen in Figure 3.4 and Table 3.2, the number of nodes does not affect the accuracy of indirect forcing approach.



(a) $N = 11$ (b) $N = 41$

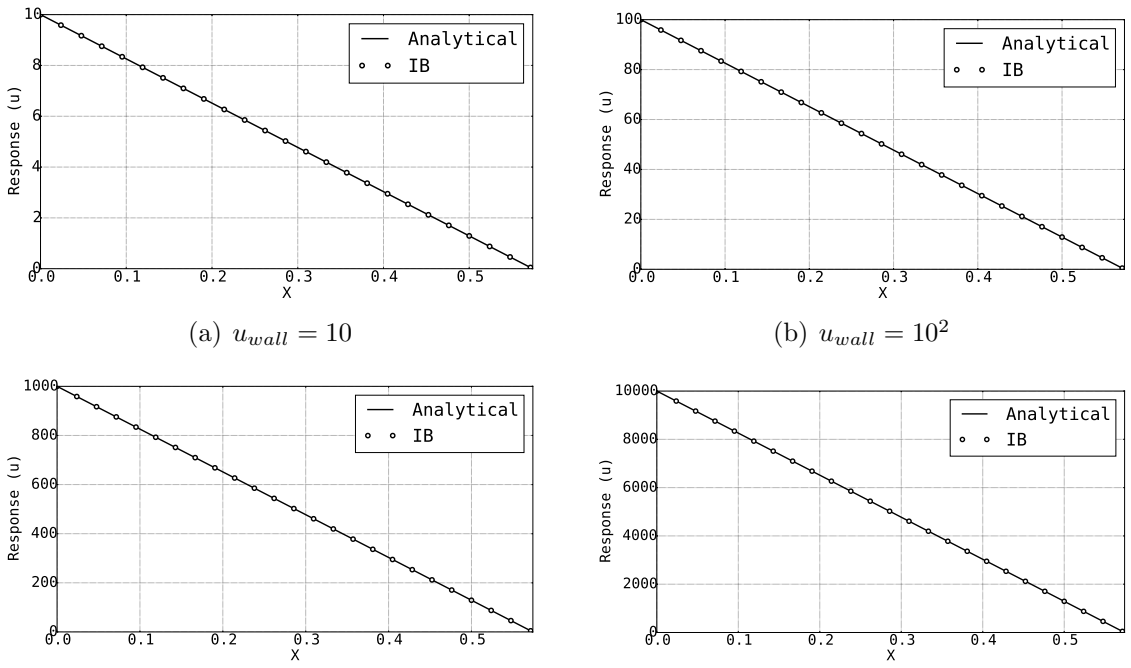
(c) $N = 81$ (d) $N = 161$

Figure 3.4: Comparison between IB and analytical results for different number of nodes.

Number of nodes	RMSE value
11	1.20×10^{-15}
41	1.50×10^{-15}
81	5.64×10^{-15}
161	3.76×10^{-15}

Table 3.2: RMSE value for different number of nodes.

For the final case, we looked at the effect of wall velocity on the accuracy of the simulation accuracy. The number of nodes is fixed at 41 and the stationary wall is located at 0.5741. The wall velocity is chosen as $10m/s$, $100m/s$, $1000m/s$, and $10000m/s$. As can be seen in Figure 3.5 and Table 3.3, the wall velocity does not affect the accuracy of the indirect forcing approach.



(a) $u_{wall} = 10$

(b) $u_{wall} = 10^2$

(c) $u_{wall} = 10^3$

(d) $u_{wall} = 10^4$

Figure 3.5: Comparison between IB and analytical results for different wall velocities.

Wall velocity	RMSE value
10	1.49×10^{-14}
100	1.52×10^{-14}
1000	1.45×10^{-14}
10000	1.48×10^{-14}

Table 3.3: RMSE value for different wall velocities.

3.5.2 Direct forcing method

In the direct forcing method, the conditions on the immersed boundary are imposed through a set of ghost cells. These are the cells in the solid region that have at least one neighbour in the fluid domain. For example, the node ϕ in Figure 3.6 is a ghost cell.

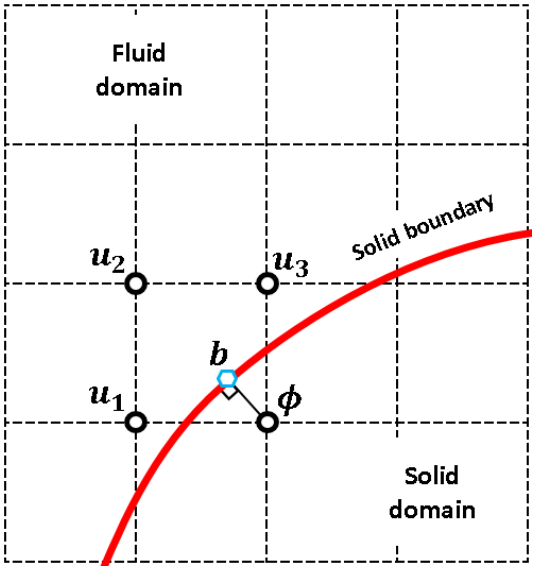


Figure 3.6: Nodes representation in the vicinity of an immersed boundary used in the ghost-cell approach.

For each ghost cell, an interpolation is used to implicitly satisfy the desired value of response on the immersed boundary. The easiest interpolation used is the bilinear interpolation (trilinear in 3D) where the value of ϕ can be defined as

$$\phi = C_1 x_1 x_2 + C_2 x_1 + C_3 x_2 + C_4 \tag{3.8}$$

where the four coefficients in Equation (3.8) are calculated based on the values of flow responses at U_1, U_2, and U_3 and the known value at the boundary point, b. Boundary points b, is the normal intercept from the ghost node to the immersed boundary. The linear interpolation is well suited for laminar, and high Reynolds number flows where the grid points are located in the viscous sublayer [50]. For high Reynolds number where higher accuracy is required, higher-order interpolations are used. Majumdat et al suggest an example of such interpolations., where they employed a linear interpolation in the tangential direction and quadratic in the normal direction [77].

In the ghost cell method, the boundary conditions are introduced on the immersed boundary surface directly in the discrete equations. Therefore, the forcing term depends on the discretization process, and its practical implementation is not straightforward compared to the continuous forcing approach that is discussed in the next section. The steps for implementing the ghost cell method can be summarized as follows

1. Mark computational nodes as fluids, solids, and ghost cells based on their relative location to the boundary
2. For each of ghost cells, define an interpolation scheme based on the neighbouring nodes
3. Update the discretized equations for nodes with ghost cells in their boundaries using the interpolation scheme
4. Solve the resulting equations for the response
5. Update the interpolation scheme and discrete equations based on the new values
6. Iterate until convergence is achieved

The ghost cell method is applied to the benchmark problem introduced in Section 3.3. We chose linear interpolation for this problem since the velocities are rather small. We considered the effect of moving wall velocity and the number of nodes on the accuracy on the solution. The IB results are verified using the analytical solution. We derive the ghost cell method, by discretizing the domain using six nodes using central difference method. For demonstration we define the wall location between nodes 2 and 3 as shown in Figure 3.7. This makes node 3 the ghost cell for this problem.

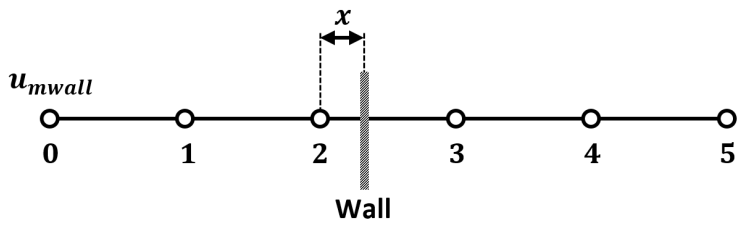


Figure 3.7: Discretized domain for the ghost cell IB method where the "wall" is represented using hashed box.

The solid wall is defined at location x in the local coordinate of the space between nodes 2 and 3. By using this coordinate and linear interpolation between nodes 2 and 3, the response on the wall is defined as

$$xu_3 + (1 - x)u_2 = u_{swall} \tag{3.9}$$

Where u_i is the response at node i and u_{swall} is the predefined velocity at the stationary wall. For this problem, we know that u_{swall} is equal to zero. Therefore, the response at the ghost cell, u_3, needs to be equal to the following to ensure zero velocity on the fixed wall.

$$u_3 = -\frac{1-x}{x}u_2 \tag{3.10}$$

The second-order central-difference differential operator for this problem is shown in Equation (3.11a). By substituting Equation (3.10) in Equation (3.11a), we modify the differential operator in a way that it incorporates the effect of solid boundary. This modified operator is shown in Equation (3.11b). It should be noted that since the value of u_3 is known from Equation (3.10), its corresponding equation is removed from Equation (3.11a).

$$L = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix} \quad \text{:original operator} \tag{3.11a}$$

$$L' = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2-(1-x)/x & 0 & 0 & 0 \\ 0 & 0 & 1+2(1-x)/x & 0 & 1 & 0 \\ 0 & 0 & -(1-x)/x & 0 & -2 & 1 \end{bmatrix} \quad \text{:modified operator} \tag{3.11b}$$

The differential operator of Equation (3.11b) needs to be updated based on the new location of the wall. This may cause a different column to become zero. As a result, we need to know the discretization technique used in solving the governing equations to implement the ghost cell method. This is not usually possible using commercial software.

The Euler method is used for time integration of the discrete equations. In the discrete form, this is written as shown in Equation (3.12a). The time integration is also needed to be modified to incorporate the effect of the ghost cell. This is done by including the known value

of ghost cell in the time integration as shown in Equation (3.12b).

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1^{n} \\ u_2^{n} \\ u_3^{n} \\ u_4^{n} \end{bmatrix} + \Delta t L \mathbf{u}^n \tag{3.12a}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -(1-x)/x & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1^{n} \\ u_2^{n} \\ u_3^{n} \\ u_4^{n} \end{bmatrix} + \Delta t L' \mathbf{u}^n \tag{3.12b}$$

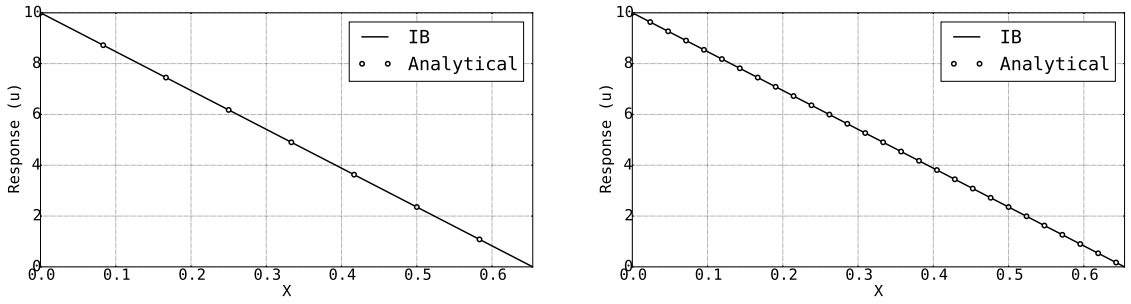
where L' is the modified differential operator of Equation (3.11b). It is clear that excessive modification of the discrete solver is required for implementation of the immersed boundary. This discretization method is applied to the benchmark case, where the effect of the number of nodes, location of the stationary wall, and the wall velocity on the solution accuracy is investigated.

For the first case, we look at the effect of the number of nodes. We choose the length of domain as $1m$ with the wall located at $x_{wall} = 0.6541$. The time step is selected as 0.1 with the moving wall velocity as $10m/s$. The number of nodes are selected as 11, 41, 81, and 161. We compared the accuracy of the solution by comparing it to the analytical result. We chose the normalized root mean square error to compared the analytical and IB results.

$$NRMSE = \frac{\sqrt{\frac{\sum_{n=1} N \left(\hat{y}_n - y\right)^2}{n}}}{y_{max} - y_{min}}$$

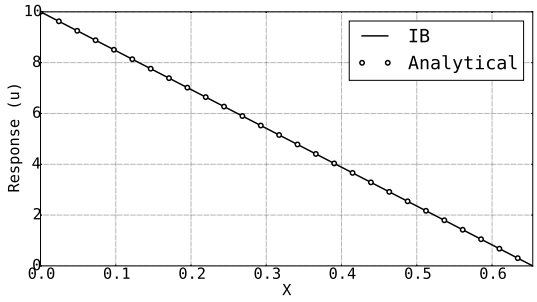
Where $\hat{y}_t$ is the predicted value, y is the actual value, and n is the number data points. Normalizing by $y_{max} - y_{min}$ enables us to compare models with different scales. This is especially useful when compared the results for different wall velocities.

As shown in Figure 3.8 and Table 3.4, the number of nodes does not affect the solution accuracy. It is possible to get good accuracies even with a low number of nodes. This is because the boundary condition is exactly satisfied at the wall location due to the ghost cell.

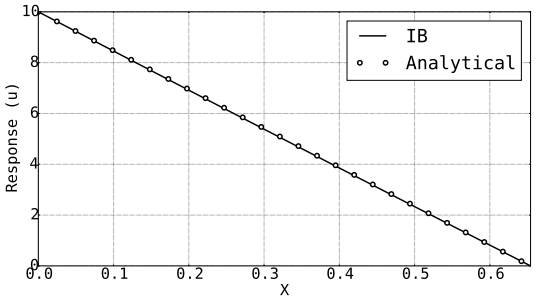


(a) N = 11

(b) N = 41



(c) N = 81



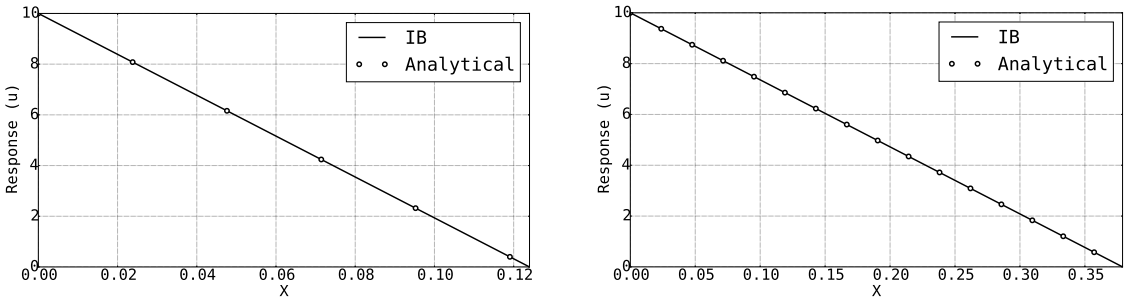
(d) N = 161

Figure 3.8: Comparison between IB and analytical results for different number of nodes.

Number of nodes	RMSE value
11	1.49×10^{-15}
41	2.08×10^{-15}
81	4.19×10^{-10}
161	9.94×10^{-10}

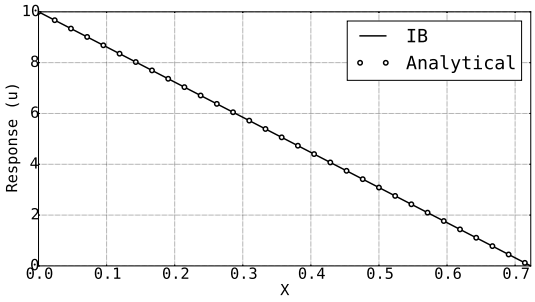
Table 3.4: RMSE value for different number of nodes.

We solved the same problem by fixing the number of nodes as 41 and changing the location of fixed bar to see its effect of the solution. The wall is located at 0.124, 0.379, 0.723, 0.936 where none of these locations coincide with the computational nodes. As shown in Figure 3.9 and Table 3.5, the solution accuracy is not affected by wall location as well.

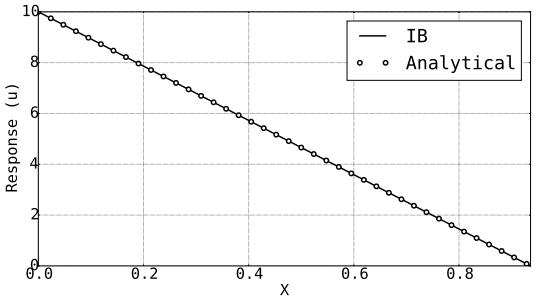


(a) $x_{wall} = 0.124$

(b) $x_{wall} = 0.379$



(c) $x_{wall} = 0.723$



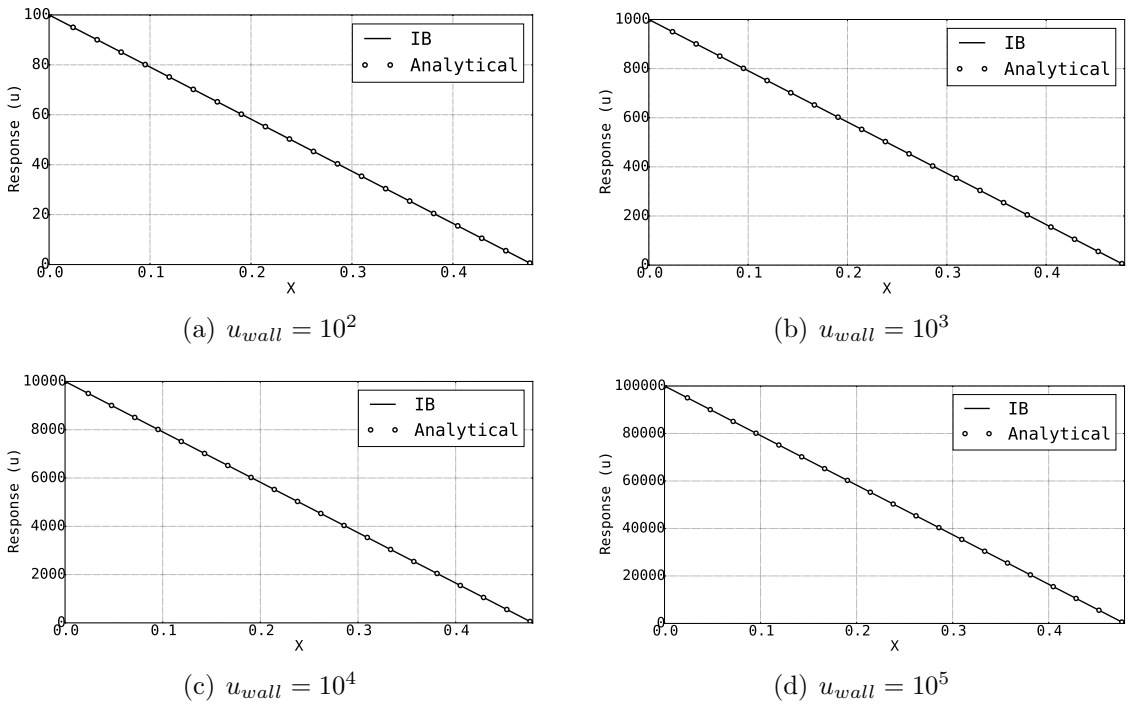
(d) $x_{wall} = 0.936$

Figure 3.9: Comparison between IB and analytical results for different wall locations.

Wall location	RMSE value
0.124	5.49E-17
0.379	3.16E-15
0.723	2.72E-14
0.936	4.99E-14

Table 3.5: RMSE value for different wall locations.

In the final investigation, we looked at the effect of wall velocity of the solution accuracy. For this case, the number of nodes is fixed as 41 and the stationary wall is located at $x_{wall} = 0.479$. The moving wall velocity is selected as 100, 1000, 10000, and 100000. As shown in Figure 3.10 and Table 3.6, the wall velocity does not affect the accuracy of the solution.



(a) $u_{wall} = 10^2$ (b) $u_{wall} = 10^3$

(c) $u_{wall} = 10^4$ (d) $u_{wall} = 10^5$

Figure 3.10: Comparison between IB and analytical results for different wall velocities.

Wall velocity	RMSE value
100	8.89×10^{-15}
1000	8.36×10^{-15}
10000	8.31×10^{-15}
100000	8.62×10^{-15}

Table 3.6: RMSE value for different wall velocities.

3.6 Continuum Forcing Method

In this implementation of the immersed boundary, a forcing equation is added to the continuous governing equation (3.1b) to represent the effect of the boundary. The continuous IB technique is the original method developed by Peskin [78] for coupled simulation of blood flow due to the contraction of heart muscle. In this approach, the immersed boundary is represented by a set of elastic fibers that their locations are tracked in a Lagrangian fashion by a collection of massless points. These points move with the local fluid velocity. Therefore, the location of the k-th Lagrangian point, X_k is governed by the following equation

$$\frac{\partial X_k}{\partial t} = u(X_k, t) \tag{3.13}$$

Where u is the velocity of the fluid at location X_k. The location of fluid nodes, x, does not necessarily coincide with the location of the Lagrangian points. Thus, it is required to map the velocities from the Eulerian domain, where the fluid's equations of motion are solved, to Lagrangian nodes. In a pure continuum problem, this can be done using the Dirac delta functions. The property of the Dirac delta function that enables the mapping between the Euler and Lagrangian domains is shown in the following equation

$$\int_{\Omega_f} f(x)\delta(x - X_k) = f(X_k) \tag{3.14}$$

As can be seen here, by convoluting the function of interest and the delta function, we can evaluate our function of interest at any location where the delta function is defined. Although this is the central idea of mapping data to the Lagrangian domain, this approach becomes unstable in practice [79]. In the practical implementation of immersed boundary method, the effect of delta function need to be expanded to a couple of nodes around X_k. This is achieved by relaxing the delta function. The relaxed delta function is referred to as regularized delta function [80]. The regularized delta function is defined using Equation (3.15) in three dimensions.

$$\delta_h(x_1, x_2, x_3) = \frac{1}{dx_1 \cdot dx_2 \cdot dx_3}\phi\left(\frac{x_1 - \eta_1}{dx_1}\right)\phi\left(\frac{x_2 - \eta_2}{dx_2}\right)\phi\left(\frac{x_2 - \eta_3}{dx_3}\right) \tag{3.15}$$

where x_i is the spatial coordinate in each direction, η_i is the location where the delta function is defined, and dx_i is the grid size in each direction. In above equation i can be 1, 2, or 3. The ϕ function is defined in Equation (3.16). As shown here there are different ways of defining this function. The input of ϕ function is r which is defined as $(x_i - \eta_i)/dx_i$.

$$\phi(r) = \begin{cases} 1 - |r| & |r| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.16a}$$

$$\phi(r) = \begin{cases} \frac{1}{3}\left(1 + \sqrt{-3r^2 + 1}\right) & |r| \leq 0.5 \\ \frac{1}{6}\left(5 - 3|r| - \sqrt{-3(1 - |r|)^2 + 1}\right) & 0.5 \geq |r| \leq 1.5 \\ 0 & \text{otherwise} \end{cases} \tag{3.16b}$$

$$\phi(r) = \begin{cases} \frac{1}{8}\left(3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}\right) & 0 \le |r| \le 1 \\ \frac{1}{8}\left(5 - 2|r| + \sqrt{-7 + 12|r| - 4r^2}\right) & 1 \le |r| \le 2 \\ 0 & \text{otherwise} \end{cases} \tag{3.16c}$$

$$\phi(r) = \begin{cases} \frac{61}{112} - \frac{11}{42}|r| - \frac{11}{56}|r|^2 + \frac{1}{12}|r|^3 + \frac{\sqrt{3}}{336}\left(243 + 1584|r|\right. & \\ \left. -748|r|^2 - 1560|r|^3 + 500|r|^4 + 336|r|^5 - 112|r|^6\right)^{1/2} & 0 \le |r| \le 1 \\ \frac{21}{16} + \frac{7}{12}|r| - \frac{7}{8}|r|^2 + \frac{1}{6}|r|^3 - \frac{3}{2}\phi\left(|r| - 1\right) & 1 \le |r| \le 2 \\ \frac{9}{8} - \frac{23}{12}|r| + \frac{3}{4}|r|^2 - \frac{1}{12}|r|^3 + \frac{1}{2}\phi\left(|r| - 2\right) & 2 \le |r| \le 3 \\ 0 & \text{otherwise} \end{cases} \tag{3.16d}$$

Equation (3.16a) is the 2-point delta function that does the linear interpolation between the points [33]. Equation (3.16b) shows the 3-point delta function used by Roma et al. [81], and Equations (3.16c) and (3.16d) are 4-point and 6-point delta functions, respectfully used by Peskin [82]. The comparison between the shape of different $\phi(r)$ functions are shown in Figure 3.11.

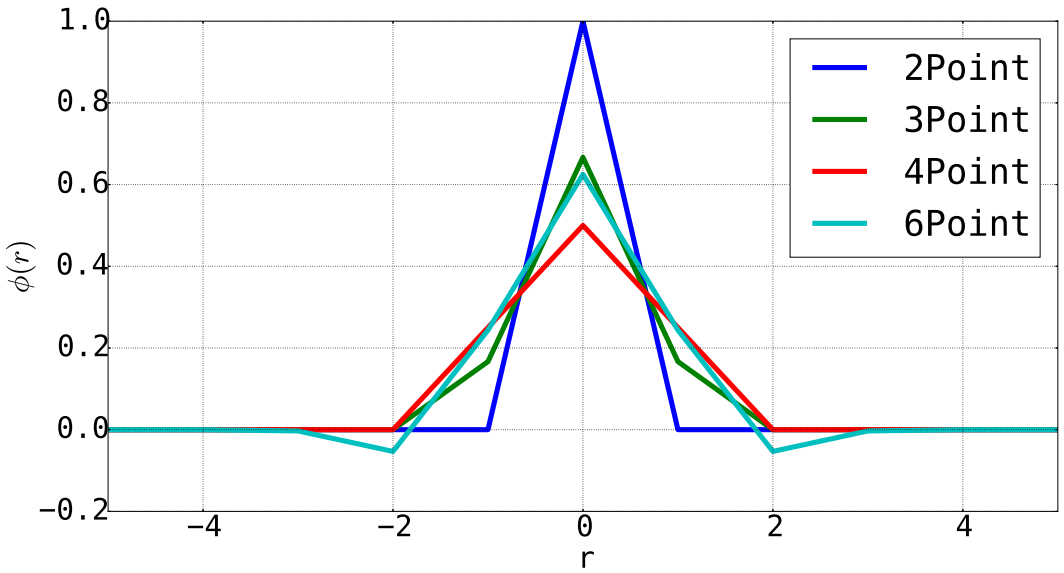


Figure 3.11: Comparison between different formulations of ϕ

Now we can write the mapping from the Eulerian nodes of the fluid solver to Lagrangian

nodes as shown in Equation (3.17).

$$u(X_k) = \int_{\Omega} u(x)\delta(x - X_k)dx \tag{3.17}$$

Where Ω is the computational domain, $u(x)$ is the velocity at the Eulerian nodes, x is the Eulerian nodes coordinate, $u(X_k)$ is the velocity at the desired Lagrangian node, and X_k is the coordinate of the Lagrangian node. Most of the continuum forcing approaches are the same up to this point. They differ depending on the way they calculate the forcing terms.

3.6.1 Classical IB method

In the classical IB method, the forces at the immersed boundaries are calculated using appropriate constitutive laws, i.e. Hooks law. This can be expressed as follows.

$$f(X_k) = \mathcal{M}(X_k) \tag{3.18}$$

Where $\mathcal{M}$ is an operator that describes the properties of the boundaries. This force is calculated at the Lagrangian points and needs to be transferred back to the Eulerian nodes. This is done using the same delta functions used previously to map the Eulerian results to Lagrangian. This method is well suited for a case of elastic bodies but will break for the cases of rigid boundaries or when the rigidity of boundaries is much more than the fluid. The origin of this problem is due to the high cycle oscillations that occur near the boundaries.

This approach is applied to the demonstration problem defined in Section 3.3. We modelled the location of the fixed boundary using the classical IB method and verified the results using analytical formulas for this problem. The forcing function is added to the right-hand-side of the governing equation as follows

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} - f(t) \tag{3.19}$$

Where $f(t)$ is the forcing function using to model the boundary. This equation is discretized using the Crank-Nicholson method as follows

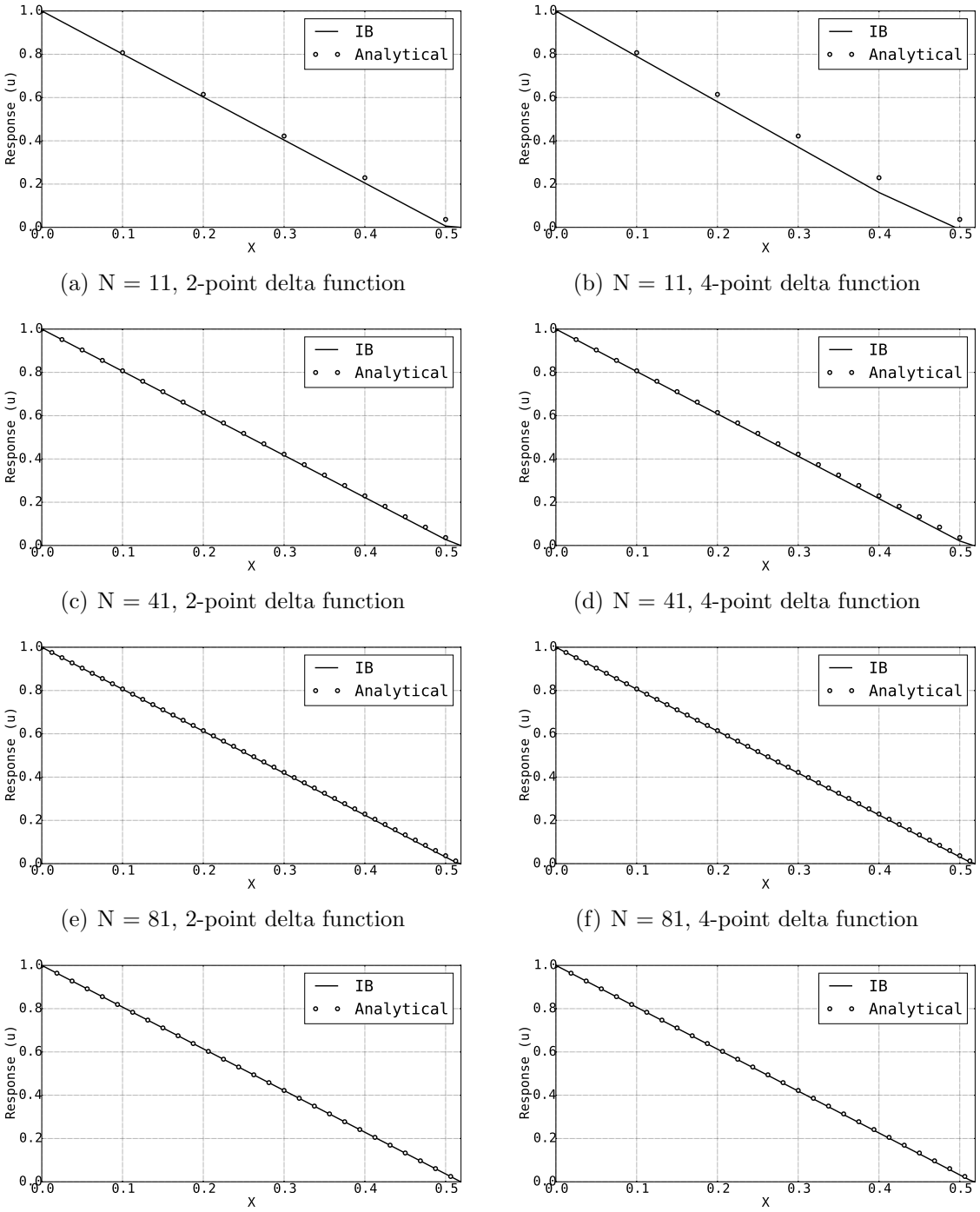
$$u^{n+1} = u^n + \frac{\Delta t}{2}\left(\frac{\partial^2 u^{n+1}}{\partial x^2} + \frac{\partial^2 u^n}{\partial x^2}\right) - \Delta t f^n \tag{3.20}$$

The steps to solve this equation and model the stationary wall using the IB method are as follows

1. Define the initial condition as u^0 and set f^0 equal to zero.
2. Calculate the velocity at the following time step, u^1, using Equation (3.20)
3. Based on the new velocity at $t = 1$, calculate the velocity at the location where the stationary wall is supposed to be, X, using δ function (U).
4. Calculate the distance that this node will move in the current time step: $X^{n+1} = X^n + U\Delta t$
5. Based on the new location of the node, calculate the forces as: $F = K\,(X^{n+1} - X^0)$, where X^0 is the desired location of the wall and K is the stiffness of wall.
6. Map the force at the Lagrangian location X to its neighbouring Eulerian points using δ function.
7. Reiterate until the convergence is satisfied.

For the classical IB method, we investigated the accuracy of the results to the number of nodes, wall velocity, and the stiffness used to model the wall. The immersed boundary simulation is compared with the analytical solution of this problem. We chose the normalized root mean square error to compared the analytical and IB results.

For the first investigation, we selected the wall stiffness as 10^2 and its location is selected as $x_{wall} = 0.51885$. We chose this location so that it does not coincide with the location of computational nodes. The time step is chosen as 10^{-5} and the wall velocity is fixed at $1m/s$. For this simulation, we used both 2-point and 4-point delta functions to transfer results between Lagrangian and Eulerian nodes. The computation domain is discretized using 11, 41, 81, and 161 nodes. The results of IB method is verified with analytical results.



(a) N = 11, 2-point delta function

(b) N = 11, 4-point delta function

(c) N = 41, 2-point delta function

(d) N = 41, 4-point delta function

(e) N = 81, 2-point delta function

(f) N = 81, 4-point delta function

(g) N = 161, 2-point delta function

(h) N = 161, 4-point delta function

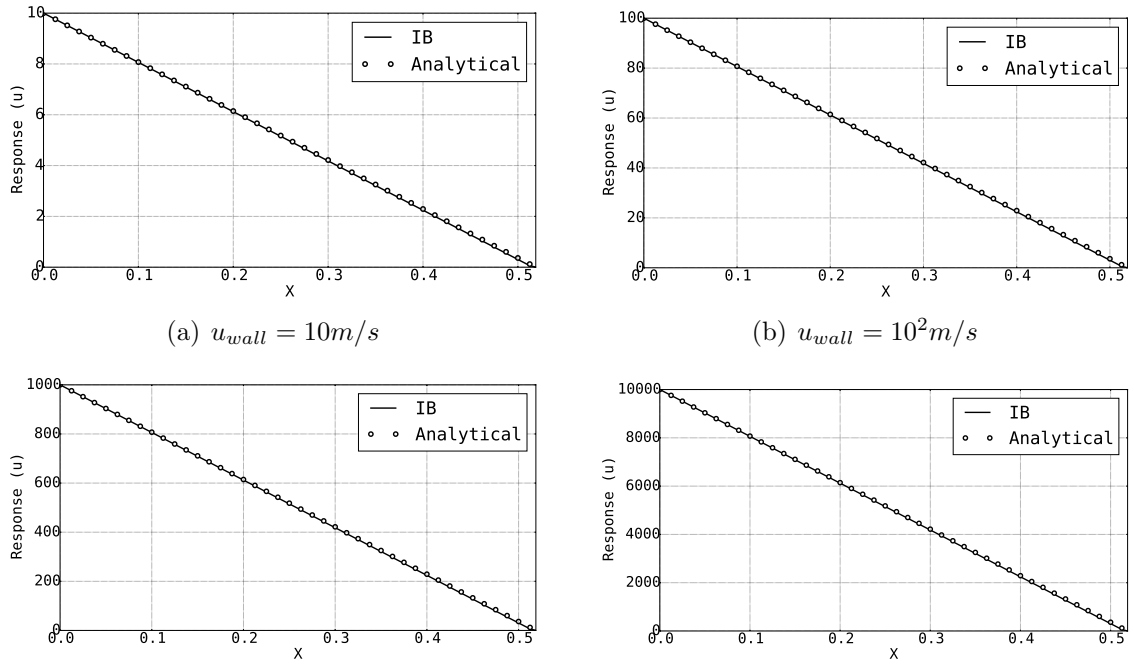
Figure 3.12: Comparison between IB and analytical results for different number of nodes and delta functions.

	RMSE value	
Number of nodes	2-point delta function	4-point delta function
11	0.0105	0.0292
41	0.0035	0.002
81	0.0026	0.0022
161	0.00023	0.0002

Table 3.7: RMSE values for different number of nodes and delta functions.

As shown in Figure 3.12 and Table 3.7, by increasing the number of nodes, the error in IB result decreases. However, changing the delta function from 2-point to 4-point does not affect the accuracy of the solution significantly. The higher order delta function can also reduce the accuracy of the solution for a very low number of nodes (here 11).

For the second investigation, we looked at the effect of changing the moving wall velocity on the solution accuracy. For this simulation, we fixed the number of nodes to 81 and set the stiffness value to 10^2. The wall location is set at 0.51885. The time step is also chosen as 10^{-5}. For this problem, we only focused on 2-point delta function. We chose the wall velocity as $10m/s$, $10^2m/s$, $10^3m/s$, and $10^4m/s$. As shown in Figure 3.13 and Table 3.8, the method is capable of handling flow with different velocity without a loss in accuracy of the results. However, the simulation time increases with increase in wall velocity.



(a) $u_{wall} = 10m/s$ (b) $u_{wall} = 10^2m/s$

(c) $u_{wall} = 10^3m/s$ (d) $u_{wall} = 10^4m/s$

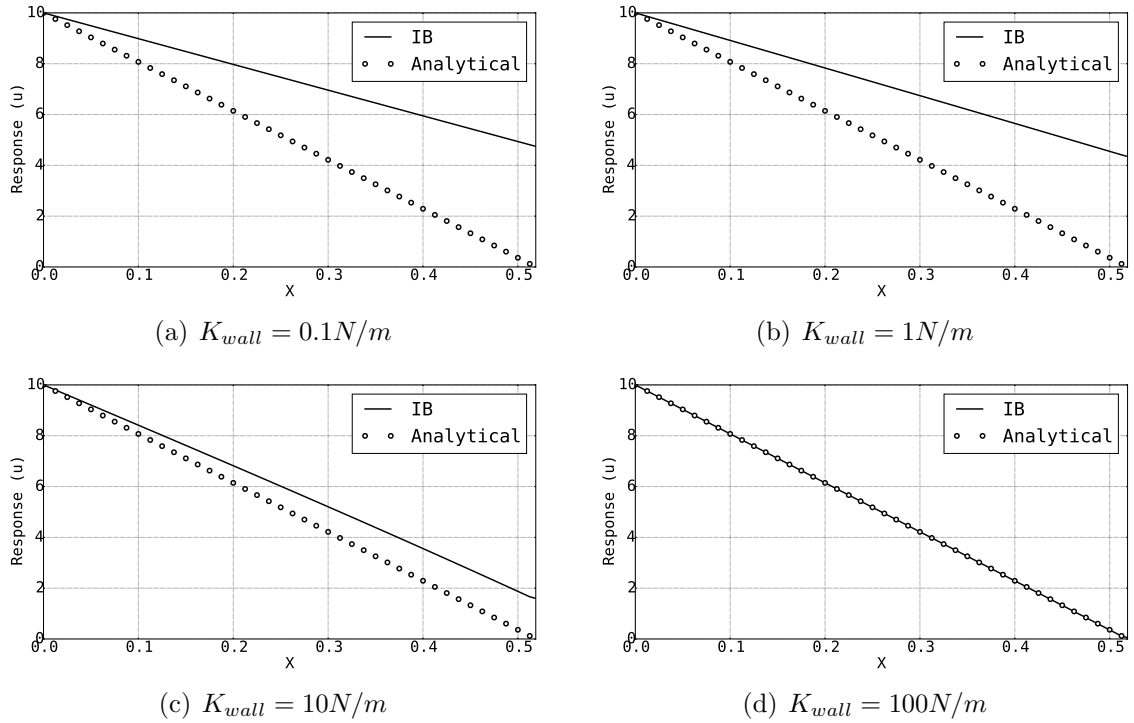
Figure 3.13: Comparison between IB and analytical results for different wall velocities.

Wall velocity (m/s)	RMSE
10	0.0024
10^2	0.0024
10^3	0.0024
10^4	0.0024

Table 3.8: RMSE values for different wall velocities.

For the last case study of classical IB method, we looked at the effect of the wall stiffness

on the accuracy of the solution. For this case, we fixed the wall location at $x_{wall} = 0.51885$ and defined the wall velocity as $10m/s$. The time step is selected as $10^{-5}s$. We looked at the wall stiffness of $0.1N/m$, $1N/m$, $10N/m$, and $100N/m$. As shown in Figure 3.14 and Table 3.9, wall stiffness has a considerable effect of the solution result. Therefore, a convergence study for the wall stiffness value is usually required if one wants to model the stiff boundaries.



(a) $K_{wall} = 0.1N/m$ (b) $K_{wall} = 1N/m$

(c) $K_{wall} = 10N/m$ (d) $K_{wall} = 100N/m$

Figure 3.14: Comparison between IB and analytical results for different wall stiffness values.

Wall stiffness (N/m)	RMSE
0.1	0.1912
1	0.1759
10	0.0665
100	0.001

Table 3.9: RMSE values for different wall stiffness values.

3.6.2 Virtual boundary method

The virtual boundary method is a different approach to imposing the effect of immersed boundary through force terms. This method is well suited for both the rigid and elastic boundaries. The difference between this approach and the classical IB method is that it does not require the solution of the constitutive equations to calculate the force terms. This method is based

on the works of the works of Goldstein et al. [41] to simulate the flow around the rigid bodies. The forcing term in the virtual boundary method is defined as

$$F(X_k, t) = \alpha \int_0^t [u(X_k, t) - U(X_k, t)]\, dt + \beta\, [u(X_k, t) - U(X_k, t)] \tag{3.21}$$

Where F is the required force at the k-th Lagrangian point, $u(X_k, t)$ is the velocity calculated from the Eulerian nodes using the δ function, and $U(X_k)$ is the desired velocity at the Lagrangian point X_k. The coefficient α and β are selected to best enforce the boundary condition at the immersed solid boundary. Equation (3.21) is essentially a way to provide a feedback control to the system to make sure that the desired velocity ($U(X_k, t)$) is achieved at the immersed boundary. From a physical standpoint, this equation represents a damped oscillator [50].

As before, we apply this technique to the demonstration problem defined in Section 3.3. We use the Crank-Nicholson method to discretize the equations as shown in the following equation.

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{1}{2}\left(\frac{\partial^2 u^{n+1}}{\partial x^2} + \frac{\partial^2 u^n}{\partial x^2}\right) \tag{3.22}$$

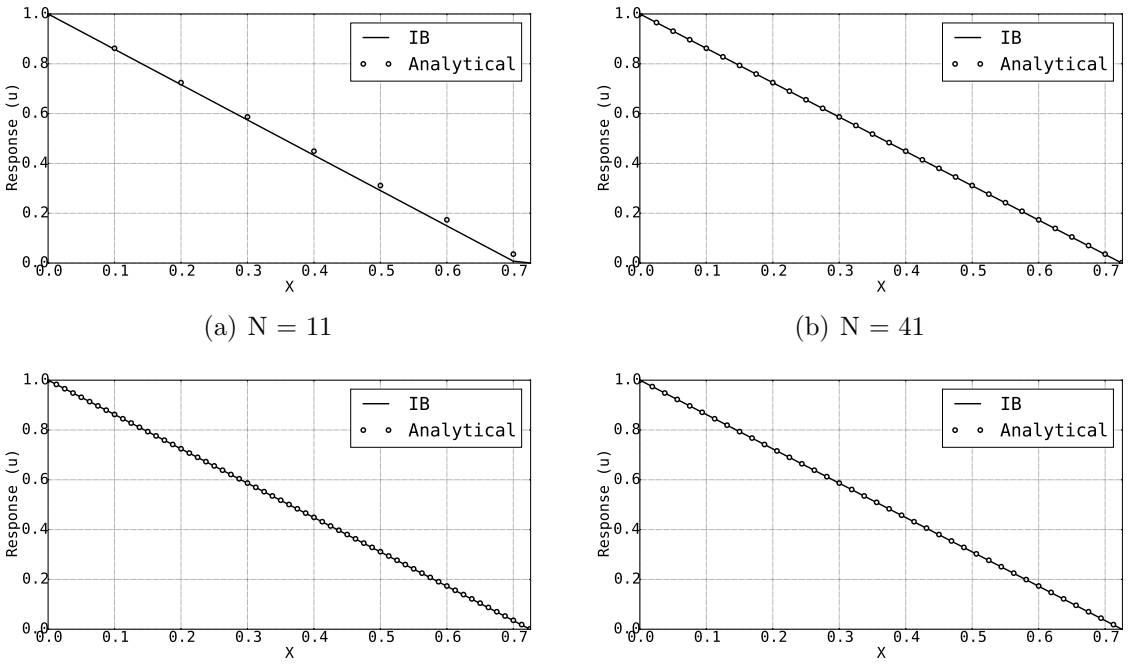
Where n is the current time step. The steps to solve this problem using the virtual boundary method can be defined as follows

1. Define the initial condition as u^0 and set f^0 equal to zero.
2. Calculate the velocity at the next time step, u^1, using Equation (3.20)
3. Map the velocity results to the Lagrangian nodes using the δ function
4. Based on the new velocity at $t = 1$ evaluate Equation (3.21).
5. Map the force at the Lagrangian location X to its neighbouring Eulerian points using δ function.
6. Reiterate until the convergence is satisfied.

Comparing the virtual boundary method to the classical IB method, we can see that fewer steps are required for this formulation. Moreover, there is no need to solve the constitutive equation for the solid domain that reduces the total computational cost.

Like the classical IB method, we investigate the effect of mesh size, moving wall velocity, and values for α and β constants of the solution accuracy. The numerical results are verified with the analytical solution of this problem. We use the RMSE value as a metric to compare the results.

We looked at the effect of mesh size on the solution accuracy of the virtual boundary method. For this case, the moving wall velocity is fixed at $10m/s$ with the location of fixed wall at $x_{wall} = 0.726$. The time step for this simulation is chosen as 10^{-5} with $\alpha = -1000$ and $\beta = -10$. We chose the number of nodes as 11, 41, 81, and 161. The total length of the domain is selected as $1m$. As shown in Figure 3.15 and Table 3.10, by increasing the nodes number the simulation results become more accurate. However, the convergence is reached earlier compared to the classical IB method of the last section. This means that the computational effort to get the same accuracy compared to classical IB method is much less when using the virtual boundary method.



(a) N = 11

(b) N = 41

(c) N = 81

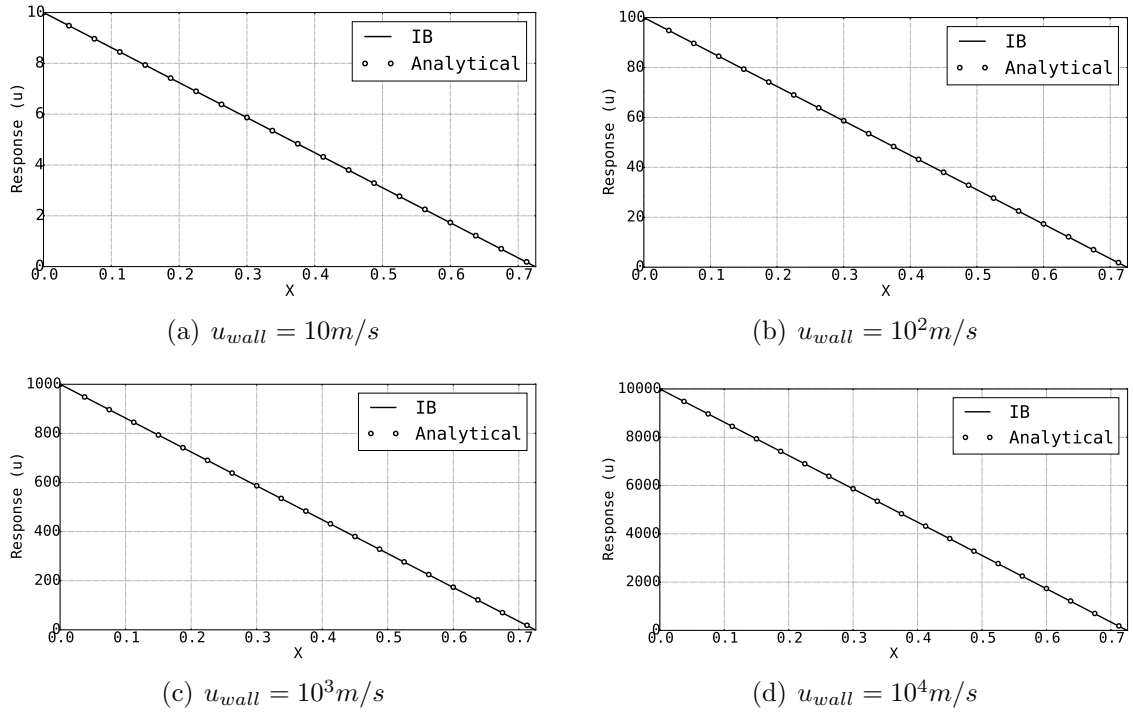
(d) N = 161

Figure 3.15: Comparison between IB and analytical results for different number of nodes.

Number of nodes	RMSE
11	0.012
41	0.00065
81	0.00063
161	0.00058

Table 3.10: RMSE values for different number of nodes.

We looked at the effect of wall velocity of the solution accuracy. As before the length of the domain is fixed at $1m$ with wall defined at $x_{wall} = 0.726$, the time step is chosen as 10^{-5}. We chose the α and β constants as -1000 and -10 respectively. We simulated the flow selecting the wall velocity as $10m/s$, $100m/s$, $1000m/s$, and $10000m/s$. The results are compared with the analytical results. As shown in Figure 3.16 and Table 3.11, the wall velocity does not affect the accuracy of the solution. When compared with the results of the classical IB method in the previous section, we can see that the virtual boundary method produces more accurate results. Moreover, the simulation time is much less than the classical IB method.



(a) $u_{wall} = 10m/s$

(b) $u_{wall} = 10^2m/s$

(c) $u_{wall} = 10^3m/s$

(d) $u_{wall} = 10^4m/s$

Figure 3.16: Comparison between IB and analytical results for different wall velocities.

Wall velocity (m/s)	RMSE
10	0.000616
10^2	0.000616
10^3	0.000616
10^4	0.000616

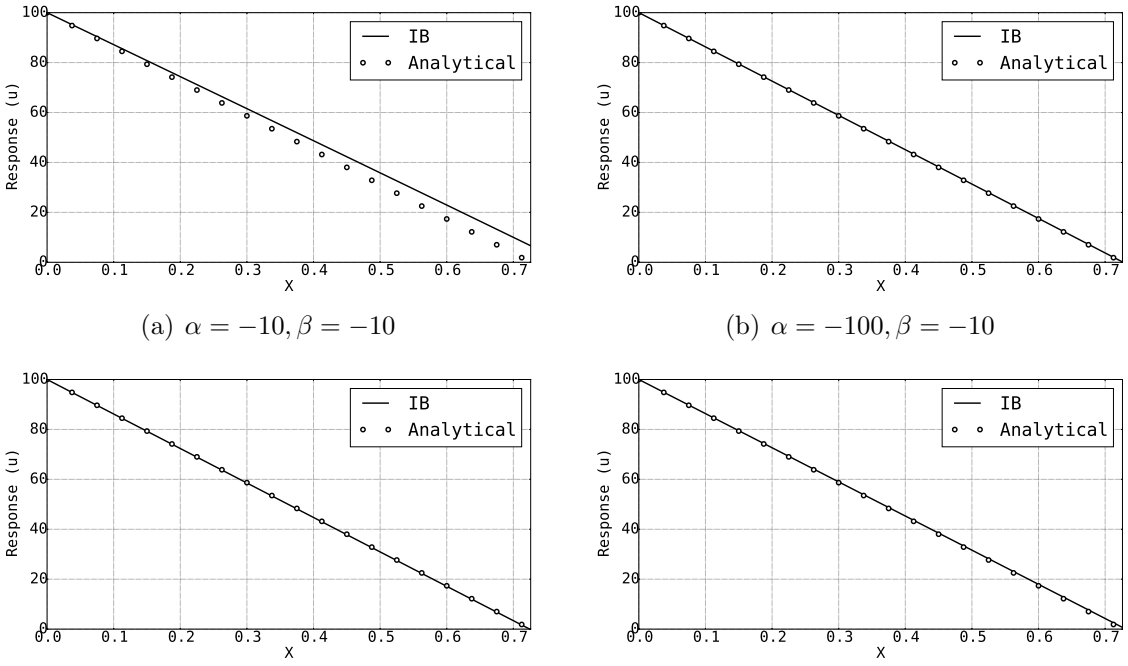
Table 3.11: RMSE values for different wall velocities.

Finally, we looked at the effect of constants α and β on the accuracy of the solution. We discretized the domain using 81 nodes with wall velocity equal to $100m/s$ with time step equal to 10^{-5}. The location of the fixed wall is chosen as $x_{wall} = 0.726$. The flow between the plates is modelled using different coefficients for α and β as shown in Table 3.12.

Case number	α	β
1	-10	-10
2	-100	-10
3	-100	-1
4	-100	-100

Table 3.12: Different values used for investigating the effect of α and β.

As can be seen in Figure 3.17 and Table 3.13, α value has a considerable effect of the accuracy of the results. Increasing the α value with an order of magnitude results in a considerable improvement in the accuracy of the solution. However, β value does not contribute to the accuracy but improves the stability of the solution. The solutions with larger β values converged in less number of iterations.



(a) $\alpha = -10, \beta = -10$

(b) $\alpha = -100, \beta = -10$

(c) $\alpha = -100, \beta = -1$

(d) $\alpha = -100, \beta = -100$

Figure 3.17: Comparison between IB and analytical results for different values for constants α and β.

α	β	RMSE
-10	-10	0.0327
-100	-10	0.00138
-100	-1	0.00129
-100	-100	0.00346

Table 3.13: Different values used for investigating the effect of α and β.

3.6.3 Penalization method

The third class of continuum immersed boundary techniques are known as penalization method that was first introduced by Arquis and Caltagirone [43]. In this method, the solid boundaries are modeled as porous media. Porosity or void fraction is a measure of the void spaces in a material and varies between 0 and 1. For the solid domain, the porosity value is near zero whereas its value is close to one in the fluid domain. Flow through a porous domain is described using the Darcy's law. This is a simple proportional relationship between the instantaneous discharge rate through a porous medium, the viscosity of the fluid and the pressure drop.

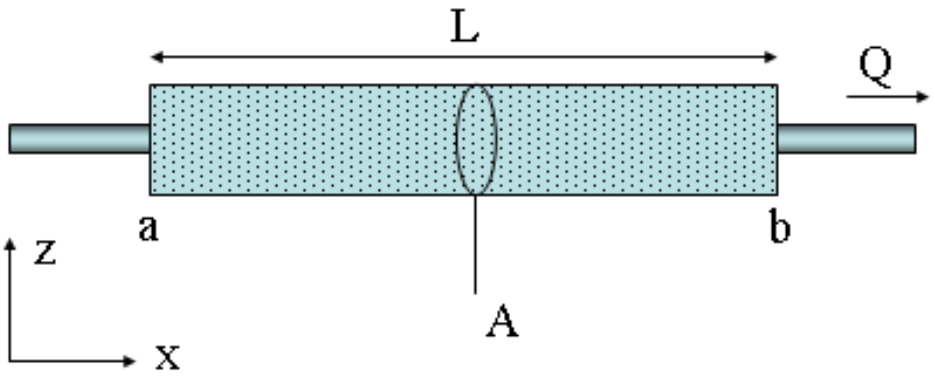


Figure 3.18: Flow through a porous pipe.

For flow through a porous domain in a pipe shown in Figure 3.18, Darcy's law is written as

$$Q = \frac{\kappa A \Delta p}{\mu L} \tag{3.23}$$

Where Q is the total flow discharge, κ is the permeability of the domain, Δp is the pressure drop due to the porosity between two ends of the pipe, μ is the fluid's viscosity, and L is the length of the domain. Darcy's law can be considered as a relation between the flow velocity and pressure drop. This pressure drop is used in the penalization method to represent the solid boundaries by modelling the force term using Equation (3.23). The force term is defined as follows

$$f = -\mathcal{H}(\mathcal{X})\frac{\mu}{\kappa}v \tag{3.24}$$

Where v is is the velocity of the flow. To apply the force term only to the region inside the solid boundary, the penalization force is multiplied by a Heaviside function, $\mathcal{H}(\mathcal{X})$, that is a function of the relative distance of the points in the domain to the boundary of the solid region. The Heaviside function $\mathcal{H}$ has the value of *one* for all points inside the solid boundary and is *zero* for points outside the boundary. This will give us a zero forcing term for points outside the solid boundary and non-zero for points inside. Therefore, the pressure drop is only applied to the points inside the solid domain. This is explained in more details in the following example.

Assume that the solid boundary is a circle, located at $(1, 2)$ with a radius of 2. This curve is defined using the following equation.

$$(x-1)^2 + (y-2)^2 = 4 \tag{3.25}$$

The relative location of an arbitrary point $x_0 = (\eta, \rho)$ to this boundary is determined using the following equation

$$\mathcal{X}(\eta, \rho) = 4 - (\eta - 1)^2 - (\rho - 2)^2 \tag{3.26}$$

Depending on the sign of $\mathcal{X}$, we can make the following conclusions

$$\begin{cases} \mathcal{X} > 0 & x_0 \text{ is inside the solid boundary} \\ \mathcal{X} < 0 & x_0 \text{ is outside the solid boundary} \\ \mathcal{X} = 0 & x_0 \text{ is on the boundary} \end{cases} \tag{3.27}$$

Sign of $\mathcal{X}$ divides the physical domain into three regions. To use this function for force term assignment to mesh cell, we need to convert its values to 0 and 1. The force terms outside the solid boundaries are multiplied by zero whereas the force terms inside the solid boundary are multiplied by one. This is done by feeding the values of function $\mathcal{X}(x)$ to a Heaviside function, $\mathcal{H}$. The Heaviside function, or the unit step function, is a discontinuous function whose value is zero for negative and one for positive values of x as shown in Figure 3.19.

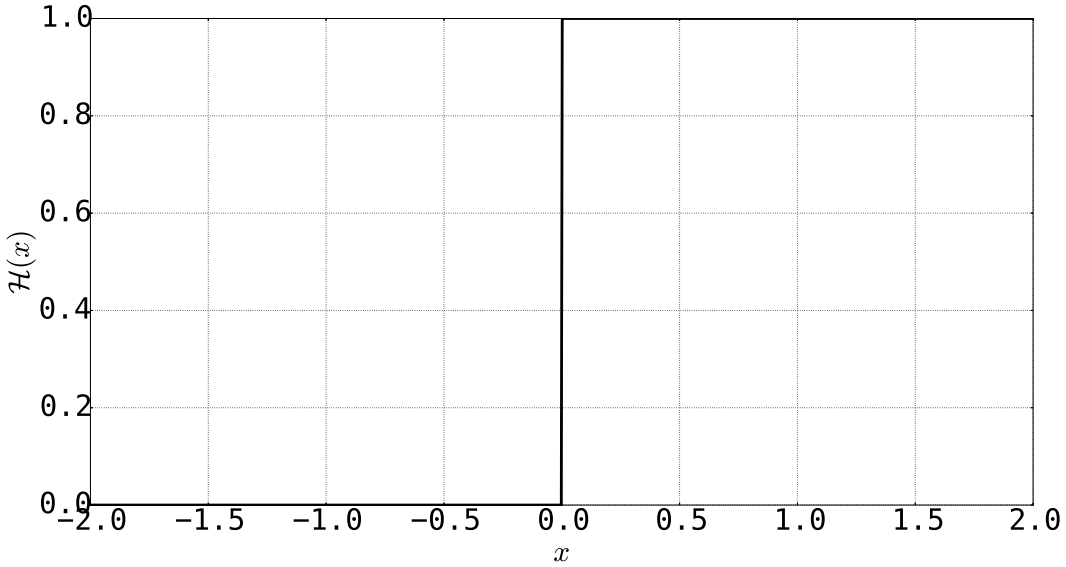
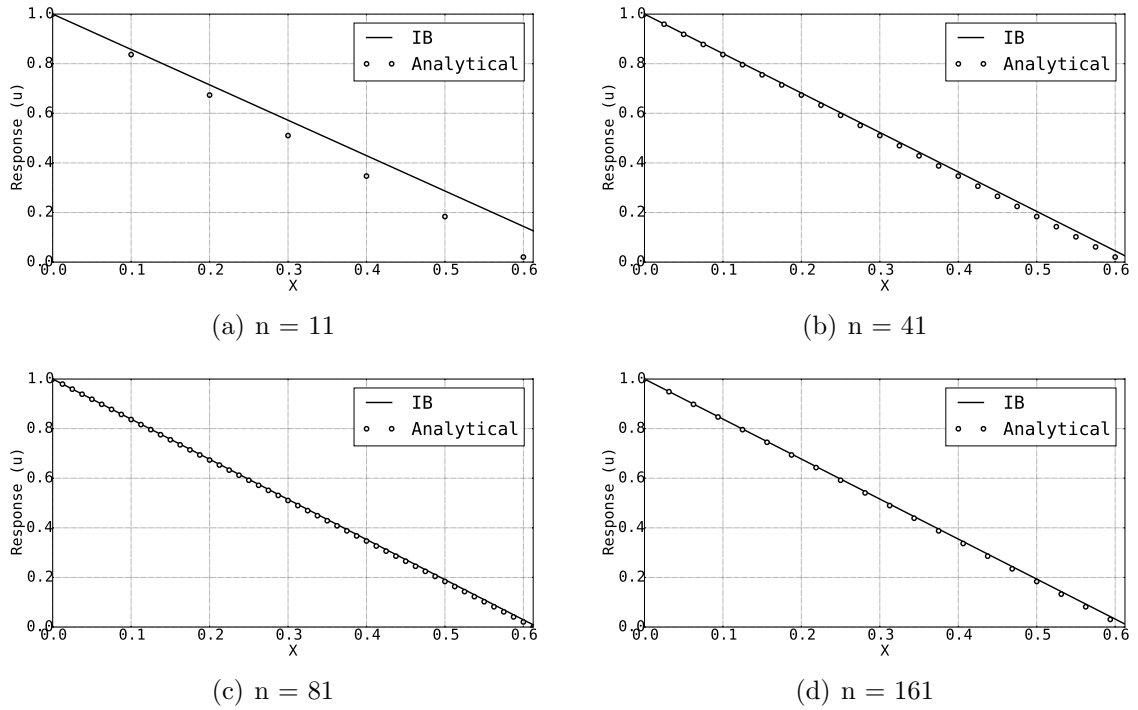


Figure 3.19: The Heaviside function.

By feeding the function $\mathcal{X}$ into the Heaviside function, we get the value of one for nodes inside the solid boundary and zero for outside. These are then multiplied to the force terms calculated using Equation (3.24). This enables us to apply the force term only within the solid domain. This method is applied to the demonstration problem of section 3.3. For this problem, we looked at the effect of the number of mesh cells, wall velocity, and the permeability value on the accuracy of the method. The results are compared with analytical results for this problem like the previous sections.

For the first set of results, we looked at the effect of mesh size on the accuracy of the solution. The domain length is chosen as $1.0m$ where the position of the fixed wall is selected as 0.6125. We chose this so that the computational nodes won't coincide the with the wall location. This better represents the application of the IB method. We chose the number of nodes as 11, 41, 81, and 161. As shown in Figure 3.20, as we increase the number of the total error between the numerical and analytical results decreases. For the Case where there is computational node exactly on top the location for the stationary wall ($n = 161$), the two results match perfectly. The IB results are shown using the solid line, and the analytical results are represented using white circles.



(a) n = 11 (b) n = 41

(c) n = 81 (d) n = 161

Figure 3.20: Comparison between IB and analytical results for different number of nodes.

For the comparison between the IB and the analytical results, we used the RSME value.

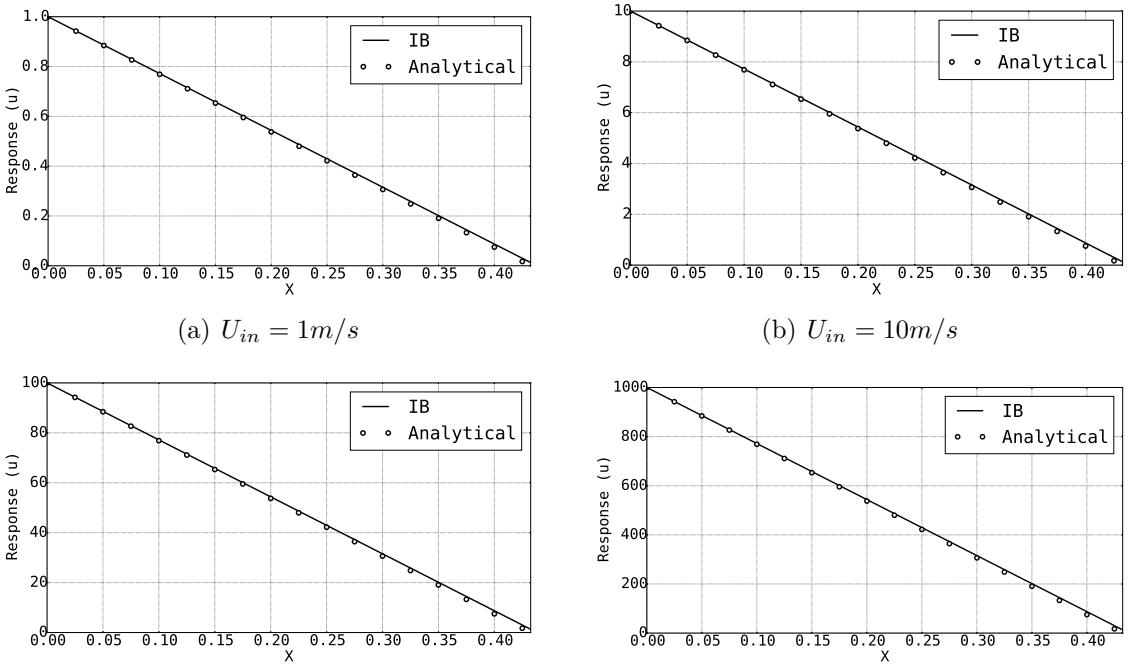
These are shown in Table 3.14.

Number of nodes	RMSE value
11	0.0624
41	0.0139
81	0.005
161	0.001

Table 3.14: RMSE values for different number of nodes.

As can be seen in Table 3.14, even for $n = 161$, the RMSE value is not zero. This is because in the penalization method, even for low values of porosity, there is still a small flow going through the solid domain.

For the subsequent investigation, we look at the effect of the different velocity of the moving wall on the accuracy of the penalization method. For this analysis, we defined the fixed wall at $x = 0.4325$ and discretized the domain using 81 nodes. The inlet velocity is selected as 1, 10, 100, and 1000. To verify the methodology, we compared the IB with analytical results. We chose the time step as 10^{-5} and the porosity value as $\kappa = 10^{-5}$. As shown in Figure 3.21 and the RMSE values in Table 3.15, the inlet velocity does not affect the accuracy of the response. Moreover, all of these simulations are done using the same time step and porosity value.



(a) $U_{in} = 1m/s$

(b) $U_{in} = 10m/s$

(c) $U_{in} = 100m/s$

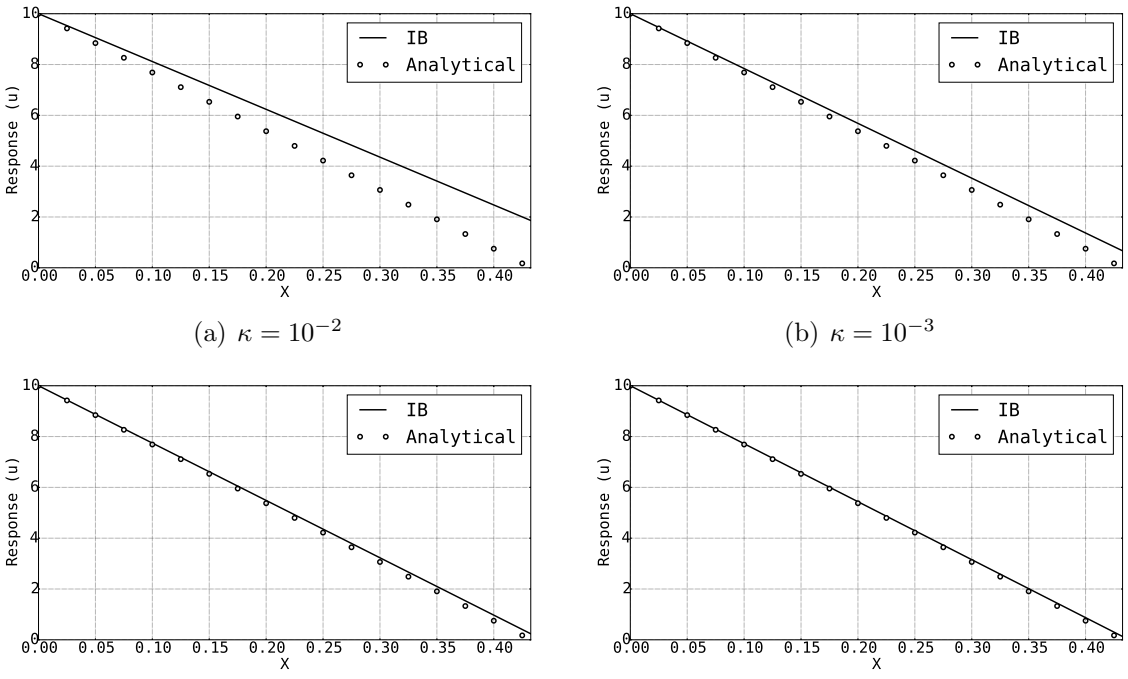
(d) $U_{in} = 1000m/s$

Figure 3.21: Comparison between IB and analytical results for different inlet velocities.

Inlet velocity (m/s)	RMSE value
1	0.0056
10	0.0056
100	0.0056
1000	0.0056

Table 3.15: RMSE values for different inlet velocities.

Finally, we looked at the effect of porosity values on the accuracy of the simulation. For this purpose, we fixed the wall location at $x_{wall} = 0.435$ and defined the velocity of the moving wall as $10m/s$. The length of the domain is selected as $1m$ and discretized using 81 nodes. We chose the time step as 10^{-5}. We investigated the accuracy of penalization method to porosity values of 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5} by comparing the IB simulation with analytical results. The results shown in Figure 3.22 and Table 3.16 displays the strong dependency of the simulation results of the porosity, κ, value. Several simulations need to be run and the convergence needs to be studied to select an appropriate porosity.



(a) $\kappa = 10^{-2}$ (b) $\kappa = 10^{-3}$

(c) $\kappa = 10^{-4}$ (d) $\kappa = 10^{-5}$

Figure 3.22: Comparison between IB and analytical results for different porosity values.

Porosity (m/s)	RMSE value
10^{-2}	0.079
10^{-3}	0.028
10^{-4}	0.01
10^{-5}	0.0056

Table 3.16: RMSE values for different porosity values.

The penalization method is probably the easiest one in the continuous IB methods to implement, however, its accuracy is very dependent on the number of nodes used. This is because there are no mechanisms to define exactly the location of solid boundaries in the penalization method. This can lead to loss of accuracy and oscillations near the boundaries. The porosity value, κ, needs to be selected by running multiple simulations are compare the results for convergence. Although even with low values of porosities, there is a leakage in the solid domain that reduces the accuracy of the simulations.

3.7 Application in Continuum Sensitivity Analysis

While considerable effort of research and development went in different techniques for modelling the flow using immersed boundary method, very few efforts were directed to calculating the sensitivity of flow with respect to change in the shape of the immersed boundary.

As mentioned in the previous chapter, there are different approaches for calculating the sensitivity response of a system. Among these methods, the discrete and continuum analytical sensitivity calculation techniques are more desired due to their accuracy and cost compared to the numerical method. However, the continuum sensitivity analysis method enables us to use a single black-box solver for the solution of both governing equation and sensitivity response. Due to this superiority, continuum sensitivity analysis is chosen as the sensitivity analysis technique. As the results, immersed boundary method of choice needs to conform to the requirements of the sensitivity analysis. As mentioned in the previous chapter, continuum sensitivity equations are derived by differentiating the continuous governing equations. Only the IB methods with continuum forcing approaches have the boundary formulation in their continuous formulation. Therefore, continuous IB is used. The discrete immersed boundary method can also be utilized if discrete sensitivity analysis is chosen. However, we do not investigate those techniques in this work. This is investigated in the works of [83, 55, 56], where they used the penalization

technique alongside with discrete sensitivity analysis method to calculate the sensitivity of flow to boundary shape changes. These works are mainly focused on Stokes flow where the Reynolds number is relatively small.

The continuum sensitivity analysis requires the governing equations be continuously differentiable. However, the delta functions used for in classical IB and virtual boundary method do not satisfy this condition. The Heaviside function employed in the penalization technique also does not satisfy this condition since a discrete step function is used as the Heaviside function to assign the force terms. This discontinuity needs to be addressed when using the CSA to calculated the sensitivities. This is achieved by introducing new regularized delta function and using regularized Heaviside functions to assign force terms to computational nodes. This is investigated in more details in the next Chapter.

3.8 Summary

In this chapter, we investigated different immersed boundary techniques in detail and compared them concerning accuracy and ease of implementation. The discrete immersed boundary techniques are more general and do not require user defined parameters to represent the immersed boundaries. However, they cannot be implemented without a thorough knowledge of discretization technique used for the governing equations. Moreover, the implementation of interpolation function used for force calculation for discrete IB can become extremely complicated for complex 3D boundaries. On the other hand, the continuum immersed boundary technique is straightforward and can be easily implemented without a deep understanding of discretization and solution procedure since the immersed boundary is handled in the continuum form of the equations. Using the continuum immersed boundary technique, it is possible to reach the same accuracy as the discrete method with the aid of tuning parameters. Most importantly, the continuum sensitivity analysis can be used in conjunction with continuum sensitivity analysis. This enables us to use the same solver for both solving the governing equations and the sensitivity analysis. Therefore, the continuum immersed boundary is used to deriving the sensitivity response of coupled aero-structural system. The current bottleneck in doing the sensitivity analysis using continuum immersed boundary method is the discontinuous mapping functions that are using to assign the forcing terms to mesh cells. This needs to be done in a

continuum fashion so that continuum sensitivity analysis can be used.

Chapter 4

Shape Sensitivity Analysis using Immersed Boundary Method

In this chapter we apply the continuum sensitivity analysis to the CFD simulations conducted using the Immersed Boundary (IB) method. In this chapter the focus is on calculating the sensitivity of flow variables such as pressure and velocity to the design variable that controls the shape of the boundary, i.e. radius of a cylinder, camber of an airfoil. The boundary of the solid domain is represented using an analytical function. However, as will be shown in the following chapters this is not a required for this method. In this chapter the modifications on the IB approach to make it suitable for continuum sensitivity formulation is discussed. To the best of the author's knowledge, this is the first time that sensitivity analysis is done for CFD simulations based the continuum IB formulation.

In contrast to traditional body conformal methods, in IB approach the solid boundaries are represented by modifying the governing equation near the boundaries. In the case of continuum IB method, this is done by adding the appropriate force term to the cells adjacent to the immersed boundary. The value of these forces are calculated either using a feedback forcing [41] or a penalization function [43]. The Navier-Stokes (NS) equations are written as shown in Equation (4.1).

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla P}{\rho} + \mu \nabla^2 \mathbf{u} + \mathbf{f} \tag{4.1}$$

where $\mathbf{u}$ is the velocity vector, P is pressure, μ is the kinematic viscosity (μ/ρ), ρ is density,

and $\mathbf{f}$ is the forcing term. The NS equations are solved on a Eulerian grid that is fixed in space and the IB is defined on a separate Lagrangian grid that can move. This is shown Figure 4.1.

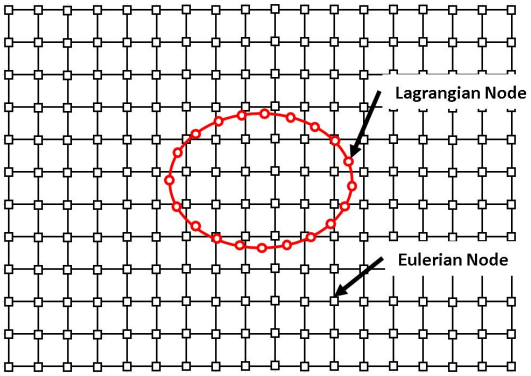


Figure 4.1: Eulerian and Lagrangian nodes for representing the fluid and solid domain. The Eulerian and Lagrangian nodes are represented by black squares and red circles.

The forcing function in the penalization method is calculated at the *Eulerian* nodes and is applied to computational nodes inside the solid boundary using a step function as shown in Equation (4.2).

$$\mathbf{f} = -\mathcal{S}(\mathcal{X}(b))\kappa\mathbf{u} \tag{4.2}$$

where $\mathcal{S}$ is the step function, $\mathcal{X}$ defines the relative location of locations inside the domain with respect to the IB boundary, κ is the penalization parameter, and $\mathbf{u}$ is the fluid velocity. Using this forcing function, the governing equation (4.1) is rewritten as shown in Equation (4.3)

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{\nabla P}{\rho} + \mu\nabla^2\mathbf{u} - \mathcal{S}(\mathcal{X}(b))\kappa\mathbf{u} \tag{4.3}$$

As mentioned in Chapter 2, to derive the continuum sensitivity formulation the governing equation of (4.3) is differentiated with respect to the design variable, b. The step function $\mathcal{S}$ derivative is Dirac delta function with is singular and cannot be used to solving the governing equations in a numerical framework. Therefore a regularized Heaviside function, $\mathcal{H}$ will be employed instead of the step function for penalizing the governing equation. The effect of this function of the simulation results and sensitivity formulation will be discussed in the following sections.

In the virtual boundary method, the forcing function is calculated at the *Lagrangian* nodes

using Equation (4.4).

$$\mathbf{f}(\mathbf{X},t) = \alpha \int_0^t \left[\mathbf{u}(\mathbf{X},\tau) - \mathbf{V}(\mathbf{X},\tau)\right] d\tau + \beta \left[\mathbf{u}(\mathbf{X},\tau) - \mathbf{V}(\mathbf{X},\tau)\right] \tag{4.4}$$

where $\mathbf{u}(\mathbf{X},t)$ is the fluid's velocity at the Lagrangian points $\mathbf{X}$ and $\mathbf{V}(\mathbf{X},t)$ is the desired velocity at the Lagrangian point. For the no-slip boundary condition at the solid boundary, the desired velocity $\mathbf{V}(\mathbf{X},t)$ is zero at each of the Lagrangian points. The Eulerian nodes where the fluid's governing equation is solved does not necessary coincide with the Lagrangian points. Moreover, the forcing functions that are evaluated at the Lagrangian nodes need to tranfered to the Eulerian nodes where the governing equations for the fluid is solved. As mentioned in Chapter 3, different mapping functions are used for transferring data between the Lagrangian and Eulerian nodes. However, none of these functions are continuously differentiable as required for the continuum sensitivity analysis. To address this issue a regularized delta function is introduced.

4.1 Regularized Heaviside/Delta Function

The Regularized Heaviside (RH) function is a step function that is smoothed so that its derivative can be calculated numerically. The RH function is also known as sigmoid function in mathematics and is characterized as a real-valued and differentiable, having either a non-negative or non-positive first derivative. There are a pair of horizontal asymptotes as $x \to \pm\infty$. The RH function is used instead of the step function to penalized the governing equation. Several different RH functions defined in Equation (4.5) are shown in Figure 4.2.

$$\mathcal{H}_1 = \frac{1}{2} + \frac{1}{\pi}\arctan(x) \tag{4.5a}$$

$$\mathcal{H}_2 = \frac{1}{1+e^{-x}} \tag{4.5b}$$

$$\mathcal{H}_3 = e^{-e^{-x}} \tag{4.5c}$$

$$\mathcal{H}_4 = \frac{1+\tanh(x)}{2} \tag{4.5d}$$

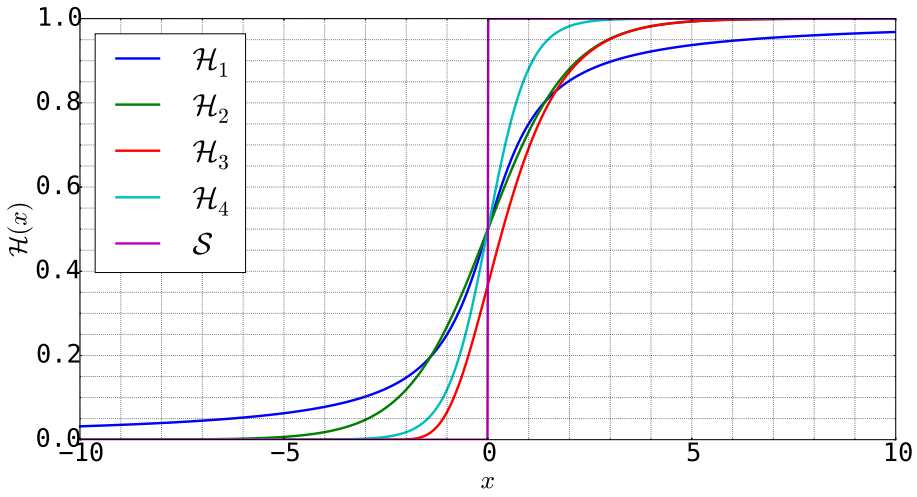


Figure 4.2: Comparison between different regularized Heaviside functions ($\mathcal{H}_i$) and step function ($\mathcal{S}$).

As shown in Figure 4.2, all the RH functions are symmetric around $x = 0$ other than $\mathcal{H}_3$. This symmetry is required since the line $x = 0$ defines the boundary of the solid domain and NS equations are penalized based on the location of this boundary. In this work, the RH function is selected as $\mathcal{H}_4 = \frac{1+\tanh(x)}{2}$ since it gives the fastest transition from 0 to 1. This transition is further controlled by adding the control parameter η to the RH function definition as shown in Equation (4.6). The effect of control parameter on the RH function is shown in Figure 4.3. As shown in Figure 4.3 the transition region is reduced by increasing the control parameter. Using this feature, the smearing effect of penalization method can be reduced near the boundaries.

$$\mathcal{H} = \frac{1 + \tanh(x/\eta)}{2} \tag{4.6}$$

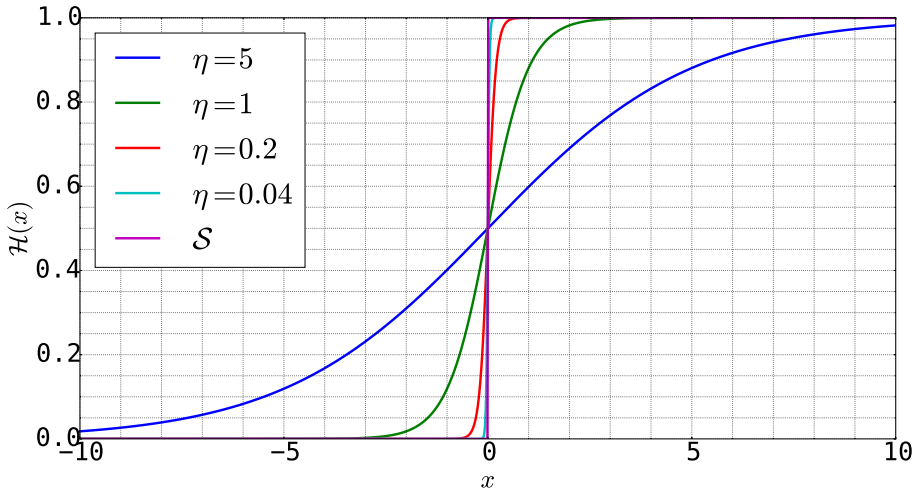


Figure 4.3: Effect of control parameter η on the RH function.

The penalized NS equation using the RH function of Equation (4.2) is continuous and can be differentiated to derive the continuum sensitivity equations. The free parameter is defined based on desired change of RH function within a specific range as shown in Equation (4.7).

$$\eta = \frac{R}{2\tanh^{-1}(p)} \tag{4.7}$$

Where R is the distance that the p percentage of the RH function occurs. For example if it is required for the RH function to change 99 percent (p = 0.99) within length of 0.01 ($R = 0.01$), the required η is calculated as 0.0018 using Equation (4.7).

For IB simulations incorporating the virtual boundary method, the Regularized Delta (RD) function is a substitute used for transferring data between the Lagrangian and Eulerian nodes. In mathematics, the Dirac delta function is defined as the derivative of the unit step function. To be consistant with the mathematical formulation, the RD function is calculated by differentiating the RH functions of Equation (4.5) as shown in Equation (4.8). These RD functions are plotted in Figure 4.4.

$$\mathcal{D}_1 = \frac{1}{\pi\left(x^2+1\right)} \tag{4.8a}$$

$$\mathcal{D}_2 = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} \tag{4.8b}$$

$$\mathcal{D}_3 = \frac{e^{-x}}{e^{e^{-x}}} \tag{4.8c}$$

$$\mathcal{D}_4 = -\frac{\tanh^2(x)+1}{2} \tag{4.8d}$$

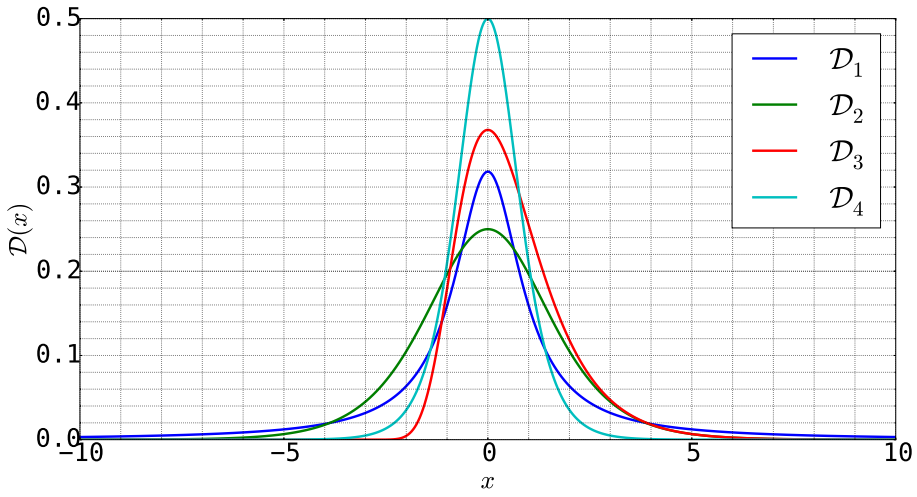


Figure 4.4: Comparison between different regularized delta functions ($\mathcal{D}_i$). The integral of all these functions over $x \in (-\infty, \infty)$ is equal to one.

RD functions $\mathcal{D}_2$ and $\mathcal{D}_3$ exhibits instability issues due to different powers of exponential in the numerator and denominator and $\mathcal{D}_1$ does not have a required sharpness expected from a RD function. Therefore, in this work, the RD function $\mathcal{D}_4$ is selected for implementation in the virtual boundary method. The sharpness of this RD function is further controlled by using the η function as used in the definition of the corresponding RH function. This is defined in Equation (4.9). The RD function for different value for η is shown in Figure 4.5.

$$\mathcal{D} = \frac{1}{\eta}\left(\frac{-\tanh^2\left(\frac{x}{\eta}\right)+1}{2}\right) \tag{4.9}$$

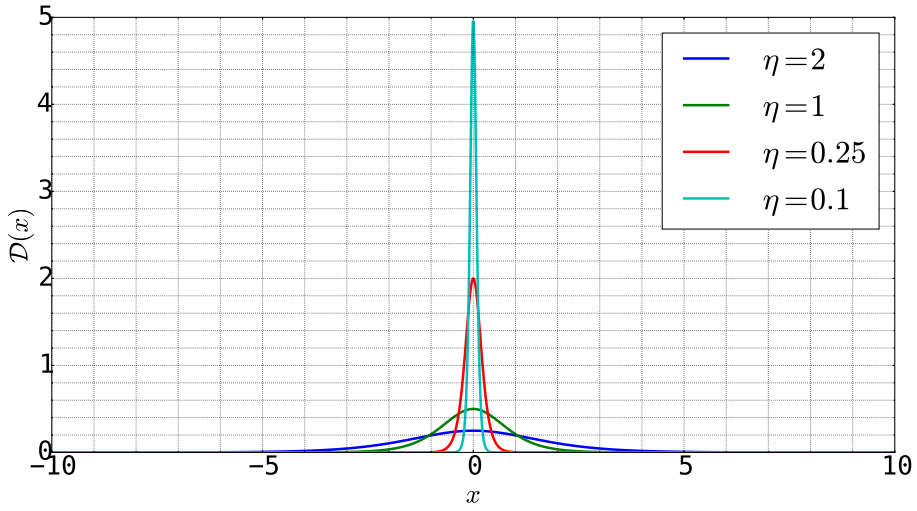


Figure 4.5: Effect of control parameter η on the RD function.

As shown in Figure 4.5, the RD function has continuous derivatives. This enables the use of virtual boundary method to be used as a part continuum sensitivity analysis for sensitivity calculation through providing differentiable governing equations.For the RD function, the free parameter is calculated by selecting how fast the RD function decay when moving further from its symmetry axis. The free parameter for the RD function is defined in Equation (4.10).

$$\eta = \frac{R}{\tanh^{-1}(\sqrt{1-p})} \tag{4.10}$$

Where R is the distance from the symmetry axis of the RD function that the value of RD function drop to p percentage of its maximum value. For example, for the RD function value to drop 99 percent ($p = 0.99$) within 0.01 ($R = 0.01$) of the symmetry axis of the RD function, the η value is calculated as 0.0033 based on Equation (4.10).

4.2 Sensitivity Analysis Formulation

In this section, the sensitivity equations are derived for IB representation of solid boundaries. Two separate IB method are selected for the sensitivity analysis, the penalization method and the virtual boundaries method. The direct and adjoint sensitivity equations for design variables that control the shape of the IB are derived for these analysis and later applied two different demonstration problems.

4.2.1 Direct Method

As mentioned in Chapter 2, the direct method for sensitivity analysis is preferred where the number of design variables are less than number of the functions that the sensitivities are calculated for. For this approach, the governing equations are differentiated with respect to the arbitrary shape design variable, b.

For the penalization method, a level-set definition of the solid boundary is used to assign penalization factors to the nodes. As defined in Chapter 3, the boundary is defined using an analytical function $\mathcal{X}$ that depends on the shape design variable. Using the RH function, the governing equation for flow over the immersed boundaries using penalization method is written

as shown in Equation (4.11).

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla P}{\rho} + \nu \nabla^2 \mathbf{u} - \mathcal{H}(\mathcal{X}(b))\kappa \mathbf{u} \tag{4.11}$$

Equation (4.11) is differentiated with respect to design variable b to derive the continuum sensitivity equations. The symbol $(\ )'$ is used instead of $\partial/\partial b$ in the sensitivity equation (4.12).

$$\frac{\partial \mathbf{u}'}{\partial t} + \mathbf{u}' \cdot \nabla \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}' = -\frac{\partial P'}{\rho} + \nu \nabla^2 \mathbf{u}' - \underbrace{\frac{\partial \mathcal{H}}{\partial \mathcal{X}} \frac{\partial \mathcal{X}}{\partial b} \kappa \mathbf{u}}_{\text{effect of shape change}} - \mathcal{H}(\mathcal{X})\kappa \mathbf{u}' \tag{4.12}$$

Equation (4.12) is solved for the velocity and pressure sensitivities ($\mathbf{u}'$, P') by knowing the values of $\mathbf{u}$ from the analysis. As shown in Equation (4.12), the effect of shape change is introduced to the sensitivity equation through the derivative of the RH function. $\partial \mathcal{H}/\partial \mathcal{X}$ is easily calculated by analytical differentiating the RH function with respect to $\mathcal{X}$. The evaluation of $\partial \mathcal{X}/\partial b$ is also straight forward since it was assumed that the solid boundary is defined analytically. The boundary conditions for Equation (4.11) are often independent of the shape design variables. Therefore, their derivative is zero. This enables us to solve the governing Equation of (4.12) for the sensitivity response.

Sensitivity implementation for the virtual boundary method is more demanding. Using the RD function, the NS equation using virtual boundary method for solid boundary definition is written as shown in Equation (4.13).

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = & -\frac{\nabla P}{\rho} + \nu \nabla^2 \mathbf{u} + \\ & \left\{ \alpha \int_0^t \left[\int \mathbf{u}(\mathbf{x}, \tau) \mathcal{D}(\mathbf{x} - \mathbf{X}) d\mathbf{x} - \mathbf{V}(\mathbf{X}, \tau) \right] d\tau + \right. \\ & \left. \beta \int \mathbf{u}(\mathbf{x}, t) \mathcal{D}(\mathbf{x} - \mathbf{X}) d\mathbf{x} - \mathbf{V}(\mathbf{X}, t) \right\} \bar{\mathcal{D}}(\mathbf{x} - \mathbf{X}) \end{aligned} \tag{4.13}$$

In above equation, $\mathbf{V}(\mathbf{X}, t)$ is the desired velocity at the Lagrangian point on the solid boundary and $\int \mathbf{u}(\mathbf{x}, t) \mathcal{D}(\mathbf{x} - \mathbf{X}) d\mathbf{x}$ is the current velocity at the Lagrangian node $\mathbf{X}$ on the solid boundary calculated from the Eulerian nodes on the fluid's domain. To transfer the forces back to the Eulerian nodes, the same normalized RD function is used as shown in Equation (4.13) by $\bar{\mathcal{D}}(\mathbf{x} - \mathbf{X})$ where the term in the parenthesis $(\mathbf{x} - \mathbf{X})$ is responsible for

casting the RD function at the location of the Lagrangian point $\mathbf{X}$. The term normalized mean that the maximum value for the RD function $\mathcal{D}$ is equal to one. This is essential to have a conservative transformation between the Eulerian and Lagrangian computational nodes. The mapping procedure is illustrated in the following example.

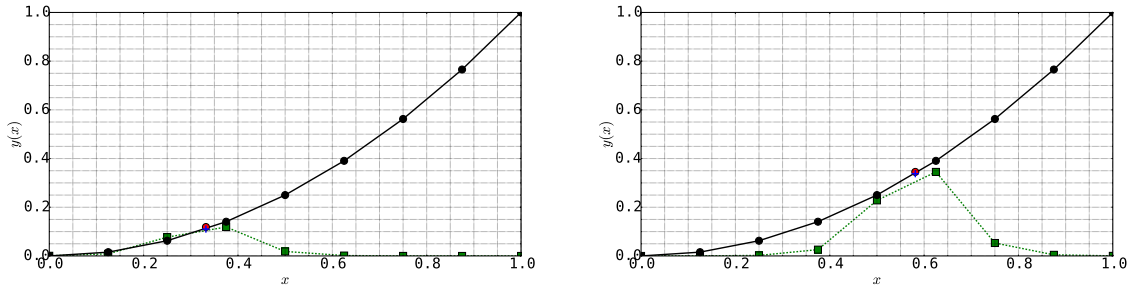
Assume that the analytical function $y = x^2$ is evaluated at 9 location between 0 and 1. To know the function value at locations $X = 0.3321$ and $X = 0.5813$ that does not coincide with the location where the function is evaluated, it is required to use the following equation. We used the RD definition of Equation (4.9) with $\eta = 0.1$ for this purpose.

$$y(X) = \sum_{i=1}^{9} \mathcal{D}_i Y_i \Delta x_i \tag{4.14}$$

where the subscript i This equation used the data from nodes around X to calculate the function value. To convert the data at X (Lagrangian node) back to the surrounding nodes where the original function was evaluated, the following equation is used.

$$y(x) = y(X)\bar{\mathcal{D}}(x - X) \tag{4.15}$$

where $\bar{\mathcal{D}}(x - X)$ is a vector of numbers. The transfer of data from Eulerian nodes to Lagrangian is shown in Figure 4.6. In this figure, the original function is represented by the solid black line and the sampling location are represented by black circles. The exact function value at the Lagrangian point is shown by red circle and the calculated value from Equation (4.14) is shown by blue cross. The numerical results for the comparison between the interpolated and actual values are shown in Table 4.1. The green squares represent the mapped back data to the Eulerian domain from the data at the Lagrangian point X shown with red circle in Figure 4.6. As shown here, the data at Lagrangian nodes mainly affects the two closest Eulerian nodes. The effect of Lagrangian data at X decreases by moving further from X. The mapping from the Lagrangian to Eulerian domain has larger error compared to the mapping from Eulerian to Lagrangian grid. However, since this is done in a loop the error diminishes as simulation moves in time.



(a) $X = 0.3321$ (b) $X = 0.5813$

Figure 4.6: Mapping the results between the Eulerian and Lagrangian computational nodes.

Lagrangian node location (X)	Absolute error
0.3321	0.0077
0.5813	0.0021

Table 4.1: Comparison between the true and interpolated result using the RD function at arbitrary Lagrangian location, X.

Using this continuous mapping between the Eulerian and Lagrangian domains, the governing equation of (4.13) is differentiated with respect to design variable b to derive the sensitivity equations. $(\,)'$ is used instead of the sensitivity operator, $\partial/\partial b$.

$$\begin{aligned}
&\frac{\partial \mathbf{u}'}{\partial t} + \mathbf{u}' \cdot \nabla \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}' = -\frac{\nabla P'}{\rho} + \nu \nabla^2 \mathbf{u}' + \\
&\quad \left\{ \alpha \int_0^t \left[\int \mathbf{u}'(\mathbf{x},\tau) \mathcal{D}(\mathbf{x}-\mathbf{X}) d\mathbf{x} + \int \mathbf{u}(\mathbf{x},\tau) \frac{\partial \mathcal{D}}{\partial X} \frac{\partial X}{\partial b} d\mathbf{x} \right] d\tau + \right. \\
&\quad \left. \beta \int \mathbf{u}'(\mathbf{x},t) \mathcal{D}(\mathbf{x}-\mathbf{X}) d\mathbf{x} + \int \mathbf{u}(\mathbf{x},t) \frac{\partial \mathcal{D}}{\partial X} \frac{\partial X}{\partial b} d\mathbf{x} \right\} \bar{\mathcal{D}}(\mathbf{x}-\mathbf{X}) + \\
&\quad \left\{ \alpha \int_0^t \left[\int \mathbf{u}(\mathbf{x},\tau) \mathcal{D}(\mathbf{x}-\mathbf{X}) d\mathbf{x} - \mathbf{V}(\mathbf{X},\tau) \right] d\tau + \right. \\
&\quad \left. \beta \int \mathbf{u}(\mathbf{x},t) \mathcal{D}(\mathbf{x}-\mathbf{X}) d\mathbf{x} - \mathbf{V}(\mathbf{X},t) \right\} \frac{\partial \bar{\mathcal{D}}}{\partial X} \frac{\partial X}{\partial b}
\end{aligned} \tag{4.16}$$

Equation (4.16) is solved for the sensitivity response $\mathbf{u}'$, where $\mathbf{u}$ is known from the analysis results. The effect of shape change is introduced in the sensitivity equation through the derivative of the RD function, $\frac{\partial D}{\partial X}\frac{\partial X}{\partial b}$. The first term in this description is easily calculated since the RD function is known. The derivative of Lagrangian node location, X, to the design variable, b, is known since the solid boundary is defined analytically. The boundary conditions of Equation (4.13) are often design independent and have zero sensitivity.

4.2.2 Adjoint Method

The adjoint method is preferred when the number of the functions the sensitivity is calculated for is less than the number of design variables. To derive the continuous adjoint sensitivity, the governing equation is defined as $\mathcal{R}(\mathbf{u}, \mathbf{b}) = 0$ and the function of which the sensitivities are calculated is defined as $h(\mathbf{u}, \mathbf{b})$. In these equations, $\mathbf{u}$ is the response variable, i.e. velocity, pressure, or displacement, and $\mathbf{b}$ is the design variable. To derive the adjoint formulation, the Lagrangian is defined as shown in Equation (4.17). It should be noted that if $\mathbf{u}$ satisfies the governing equation, the Lagrangian, Γ, is equal to the function of interest. Therefore, its sensitivity is equal to sensitivity of the Lagrangian.

$$\Gamma(\mathbf{u}, \mathbf{b}) = h(\mathbf{u}, \mathbf{b}) + \lambda \mathcal{R}(\mathbf{u}, \mathbf{b}) \tag{4.17}$$

Differentiating Equation (4.17) and rewriting the terms results in the following equation.

$$\frac{d\Gamma}{db_i} = \frac{\partial h}{\partial b_i} + \left[\frac{\partial h}{\partial \mathbf{u}} + \lambda \frac{\partial \mathcal{R}}{\partial b_i}\right] \frac{\partial \mathbf{u}}{\partial b_i} + \lambda \frac{\partial \mathcal{R}}{\partial b_i} \tag{4.18}$$

To avoid calculating $\dfrac{\partial \mathbf{u}}{\partial b_i}$, its coefficient is set equal to zero. This results in Equation (4.19) for the adjoint variable λ.

$$\lambda = -\left(\frac{\partial \mathcal{R}}{\partial \mathbf{u}}\right)^{-1} \frac{\partial h}{\partial \mathbf{u}} \tag{4.19}$$

By knowing the adjoint vector, the sensitivity of Lagrangian function, $\Gamma(\mathbf{u}, \mathbf{b})$, is written as shown in Equation (4.20)

$$\frac{d\Gamma}{db_i} = \frac{\partial h}{\partial b_i} + \lambda \frac{\partial \mathcal{R}}{\partial b_i} \tag{4.20}$$

The adjoint formulation is implemented for a simple continuous system as defined in Figure 4.7. The design variables are selected as the cross-sectional areas of the axial bars, A_1 and A_2. The length of bars are selected as L_1 and L_2 with a force applied at the tip, F. We are interested in the sensitivity of stress in bar (1) with respect to the cross-section areas A_1 and A_2.

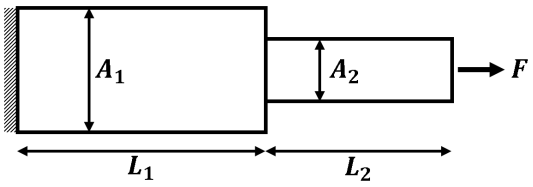


Figure 4.7: Step bar problem.

The governing equation for this problem is defined in Equation (4.21).

$$\mathcal{R}(\sigma, A_1, A_2) = \frac{\partial(\sigma A)}{\partial x} \tag{4.21}$$

The solution for stress in the domain is found by integrating Equation (4.21). However, due to discontinuity at $x = L_1$, the integration is done for each section of the domain as shown in Equation (4.22). These results agree with what is expected for the stress distribution for this problem.

$$\int_0^{x<L_1} \frac{\partial(\sigma A)}{\partial x} dx = 0 \rightarrow \sigma_x A_1 - F = 0 \rightarrow \sigma_x = \frac{F}{A_1} \tag{4.22a}$$

$$\int_{L_1}^{L_1+x} \frac{\partial(\sigma A)}{\partial x} dx = 0 \rightarrow \sigma_x A_2 - F = 0 \rightarrow \sigma_x = \frac{F}{A_2} \tag{4.22b}$$

The sensitivity of the stress in the first element to the design variables, is first calculated using the direct method. This requires solving the sensitivity equation once for each of the design variables. The sensitivity equation is written by differentiating the governing equation (4.21) as shown in Equation (4.23).

$$\frac{\partial}{\partial x}\left(\frac{\partial \sigma}{\partial A_i} A + \sigma \frac{\partial A}{\partial A_i}\right) = 0 \tag{4.23}$$

The sensitivity of stress in the first bar is calculated by integrated above equation. This needs to be done for each of the design variables as shown in Equation (4.24).

$$\int_0^x \frac{\partial}{\partial x}\left(\frac{\partial \sigma}{\partial A_1} A + \sigma \frac{\partial A}{\partial A_1}\right) dx = 0$$
$$\frac{\partial \sigma}{\partial A_1} A_1 + \sigma_1 = 0 \rightarrow \frac{\partial \sigma_1}{\partial A_1} = -\frac{\sigma_1}{A_1} \tag{4.24a}$$

$$\int_0^x \frac{\partial}{\partial x}\left(\frac{\partial \sigma}{\partial A_2}A + \sigma\frac{\partial A}{\partial A_2}\right)dx = 0$$

$$\frac{\partial \sigma}{\partial A_2}A_2 + \sigma_2 = 0 \rightarrow \frac{\partial \sigma_2}{\partial A_2} = -\frac{\sigma_2}{A_2} \tag{4.24b}$$

These results agree with the analytical sensitivity results for this problem. The Lagrangian function is defined as shown in Equation (4.25).

$$\Gamma = h(\sigma) + \mathcal{R}(\sigma, A) \tag{4.25}$$

where $h(\sigma)$ is the stress in first element, and $\mathcal{R}(\sigma, A)$ is the governing equation. $\mathcal{R}(\sigma, A)$ is equal to zero when the governing equation is satisfied therefore, calculating the sensitivity of Γ is same as h. This is done in Equation (4.26).

$$\frac{d\Gamma}{d\bar{A}} = \frac{\partial h}{\partial \bar{A}} + \lambda\frac{\partial \mathcal{R}}{\partial \bar{A}} + \left[\frac{\partial h}{\partial \bar{\sigma}} + \lambda\frac{\partial \mathcal{R}}{\partial \bar{\sigma}}\right]\frac{\partial \bar{\sigma}}{\partial \bar{A}} \tag{4.26}$$

where $\bar{}$ represents a vector. $\dfrac{\partial h}{\partial \bar{A}}$ is equal to zero since $h = \sigma_1$ and the stress does not explicitly related to the cross-sectional area. $\dfrac{\partial h}{\partial \bar{\sigma}}$ is calculated easily. $\dfrac{\partial \mathcal{R}}{\partial \bar{\sigma}}$ is calculated as shown in Equation (4.27).

$$\frac{\partial \mathcal{R}}{\partial \sigma_1} = \frac{\partial}{\partial x}\left(\frac{\partial \sigma}{\partial \sigma_1}A + \sigma\frac{\partial A}{\partial \sigma_1}\right)$$

$$\rightarrow \int_0^{L_1} \frac{\partial}{\partial x}\left(\frac{\partial \sigma_1}{\partial \sigma_1}A_1\right)dx + \int_{L_1}^{L_1+x} \frac{\partial}{\partial x}\left(\frac{\partial \sigma_2}{\partial \sigma_1}A_2\right)dx = A_1 \tag{4.27a}$$

$$\frac{\partial \mathcal{R}}{\partial \sigma_1} = \frac{\partial}{\partial x}\left(\frac{\partial \sigma}{\partial \sigma_2}A + \sigma\frac{\partial A}{\partial \sigma_2}\right)$$

$$\rightarrow \int_0^{L_1} \frac{\partial}{\partial x}\left(\frac{\partial \sigma_1}{\partial \sigma_2}A_1\right)dx + \int_{L_1}^{L_1+x} \frac{\partial}{\partial x}\left(\frac{\partial \sigma_2}{\partial \sigma_2}A_2\right)dx = A_2 \tag{4.27b}$$

Using Equation (4.27), it is possible to solve for the ajoint vector λ in Equation (4.26).

$$\frac{\partial h}{\partial \bar{\sigma}} + \lambda \frac{\partial \mathcal{R}}{\partial \bar{\sigma}} = 0$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \lambda \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \lambda = - \begin{bmatrix} 1/A_1 \\ 0 \end{bmatrix} \tag{4.28}$$

Finally, to calculate the sensitivities it is required to calculate the sensitivity of the governing equation (4.21) to the design variables. This is easy to calculate since the no solution is needed. This is done in Equation (4.29).

$$\frac{\partial \mathcal{R}}{\partial A_1} = \int_0^{x<L_1} \frac{\partial \sigma}{\partial x} dx = \sigma_1 \tag{4.29a}$$

$$\frac{\partial \mathcal{R}}{\partial A_2} = \int_{L_1}^{x<L_2} \frac{\partial \sigma}{\partial x} dx = \sigma_2 \tag{4.29b}$$

The sensitivity of stress in first bar is calculate by substituting Equations (4.28) and (4.29) into Equation (4.26). The sensitivity of stress in first bar to design variables using adjoint method is written as

$$\frac{d\Gamma}{d\bar{A}} = - \begin{bmatrix} 1/A_1 \\ 0 \end{bmatrix}^T \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} = - \begin{bmatrix} \sigma_1/A_1 \\ 0 \end{bmatrix} \tag{4.30}$$

This agrees with the analytical results of the sensitivities.

4.3 Shape Sensitivity Analysis for 1D problem

For the first benchmark case, we looked at the flow between two plates that was defined in Section 3.3. The governing equation and boundary conditions for this problem is shown in Equation (4.31). The boundary condition is defined as $u = U$ at $y = L$.

$$u_t = \mu u_{yy} \quad \text{in } \Omega_f \tag{4.31a}$$

$$\begin{cases} u = U \quad \text{at } y = L \\ u = 0 \quad \text{at } y = 0 \end{cases} \tag{4.31b}$$

The analytical result for the steady-state velocity profile between the two plates is defined in Equation (4.32)

$$u = \frac{Ux}{L} \tag{4.32}$$

This enables us to treat the moving wall velocity (U) and the distance between the two plates (L) as design variables. The sensitivity of steady-state velocity profile between the two plate to these design variables are calculated using CSA. The results of the CSA are verified with the analytical sensitivity results in Equation (4.33).

$$\frac{\partial u}{\partial L} = -\frac{Ux}{L^2} \tag{4.33a}$$

$$\frac{\partial u}{\partial U} = \frac{x}{L} \tag{4.33b}$$

For this benchmark problem, the flow is modeled using two different immersed boundary techniques introduced in the beginning of this chapter, penalization method and the virtual boundary method. This difference between the analysis of flow in this chapter and Chapter 3 is the use of RH and RD functions instead of discontinuous step and delta function. Therefore, first the effect of these modifications are investigated on the simulation results and then the sensitivity analysis is conducted.

4.3.1 Effect of RH/RD function on simulation

4.4 Shape Sensitivity of Flow over a Cylinder

4.5 Shape Sensitivity of Flow through a Nozzle

4.6 Summary

Bibliography

[1] Sridhar, B., Ng, H., and Chen, N., "Aircraft trajectory optimization and contrails avoidance in the presence of winds," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 5, 2011, pp. 1577–1584.

[2] Han, Z.-H., Görtz, S., and Zimmermann, R., "Improving variable-fidelity surrogate modeling via gradient-enhanced kriging and a generalized hybrid bridge function," *Aerospace Science and Technology*, Vol. 25, No. 1, 2013, pp. 177–189.

[3] Pettit, C. L., "Uncertainty quantification in aeroelasticity: recent results and research challenges," *Journal of Aircraft*, Vol. 41, No. 5, 2004, pp. 1217–1229.

[4] Sobieszczanski-Sobieski, J. and Haftka, R. T., "Multidisciplinary aerospace design optimization: survey of recent developments," *Structural optimization*, Vol. 14, No. 1, 1997, pp. 1–23.

[5] Martins, J. R., Sturdza, P., and Alonso, J. J., "The complex-step derivative approximation," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 29, No. 3, 2003, pp. 245–262.

[6] Naumann, U., *The art of differentiating computer programs: an introduction to algorithmic differentiation*, Vol. 24, Siam, 2012.

[7] Choi, K. K. and Kim, N.-H., *Structural sensitivity analysis and optimization 1: linear systems*, Springer Science & Business Media, 2006.

[8] Vanderplaats, G., *Numerical optimization techniques for engineering design: with applications*, McGraw-Hill Series in Mechanical Engineering, McGraw-Hill Ryerson, Limited, 1984.

[9] Arora, J. S. and Haug, E. J., "Methods of design sensitivity analysis in structural optimization," *AIAA journal*, Vol. 17, No. 9, 1979, pp. 970–974.

[10] Reuther, J. J., Jameson, A., Alonso, J. J., Rimllnger, M. J., and Saunders, D., "Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 2," *Journal of aircraft*, Vol. 36, No. 1, 1999, pp. 61–74.

[11] Martins, J. R., Alonso, J. J., and Reuther, J. J., "A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design," *Optimization and Engineering*, Vol. 6, No. 1, 2005, pp. 33–62.

[12] Newman, J. C., Taylor, A. C., and Barnwell, R. W., *Aerodynamic shape sensitivity analysis and design optimization of complex configurations using unstructured grids*, National Aeronautics and Space Administration, 1997.

[13] Morris, A., Allen, C., and Rendall, T., "CFD-based optimization of aerofoils using radial basis functions for domain element parameterization and mesh deformation," *International journal for numerical methods in fluids*, Vol. 58, No. 8, 2008, pp. 827–860.

[14] Jameson, A., Shankaran, S., Martinelli, L., and Haimes, B., "Aerodynamic Shape Optimization of Complete Aircraft Configurations using Unstructured Grids," *42nd AIAA Aerospace Sciences Meeting and Exhibit*, AIAA 2004-533, 2004.

[15] Gamboa, P., Vale, J., P. Lau, F., and Suleman, A., "Optimization of a morphing wing based on coupled aerodynamic and structural constraints," *AIAA journal*, Vol. 47, No. 9, 2009, pp. 2087–2104.

[16] Pandya, M. J. and Baysal, O., "Gradient-based aerodynamic shape optimization using alternating direction implicit method," *Journal of aircraft*, Vol. 34, No. 3, 1997, pp. 346–352.

[17] Kim, H.-J., Sasaki, D., Obayashi, S., and Nakahashi, K., "Aerodynamic optimization of supersonic transport wing using unstructured adjoint method," *AIAA journal*, Vol. 39, No. 6, 2001, pp. 1011–1020.

[18] Lyu, Z. and Martins, J. R., "Aerodynamic design optimization studies of a blended-wing-body aircraft," *Journal of Aircraft*, Vol. 51, No. 5, 2014, pp. 1604–1617.

[19] Haftka, R. T. and Adelman, H. M., "Recent developments in structural sensitivity analysis," *Structural optimization*, Vol. 1, No. 3, 1989, pp. 137–151.

[20] Borggaard, J. and Burns, J., "A PDE sensitivity equation method for optimal aerodynamic design," *Journal of Computational Physics*, Vol. 136, No. 2, 1997, pp. 366–384.

[21] Hristova, H., Etienne, S., Pelletier, D., and Borggaard, J., "A continuous sensitivity equation method for time-dependent incompressible laminar flows," *International Journal for numerical methods in fluids*, Vol. 50, No. 7, 2006, pp. 817–844.

[22] Cross, D. M. and Canfield, R. A., "Local continuum shape sensitivity with spatial gradient reconstruction," *Structural and Multidisciplinary Optimization*, Vol. 50, No. 6, 2014, pp. 975–1000.

[23] Arora, J. and Haug, E., "Methods of Design Sensitivity Analysis in Structural Optimization," *AIAA Journal*, Vol. 17, No. 9, 1979, pp. 970–974.

[24] Dems, K. and Mroz, Z., "Variational approach to first- and second-order sensitivity analysis of elastic structures," *International Journal for Numerical Methods in Engineering*, Vol. 21, No. 4, 1985, pp. 637–661.

[25] Borggaard, J. and Burns, J., *A sensitivity equation approach to shape optimization in fluid flows*, Springer, 1995.

[26] Stanley, L. G. D. and Stewart, D. L., *Design Sensitivity Analysis: Computational Issues of Sensitivity Equation Methods*, Frontiers in applied mathematics, Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2002.

[27] Etienne, S. and Pelletier, D., "A general approach to sensitivity analysis of fluid–structure interactions," *Journal of Fluids and Structures*, Vol. 21, No. 2, 2005, pp. 169–186.

[28] Liu, S. and Canfield, R. A., "Equivalence of continuum and discrete analytic sensitivity methods for nonlinear differential equations," *Structural and Multidisciplinary Optimization*, Vol. 48, No. 6, 2013, pp. 1173–1188.

[29] Sahin, M. and Mohseni, K., "An arbitrary Lagrangian–Eulerian formulation for the numerical simulation of flow patterns generated by the hydromedusa Aequorea victoria," *Journal of Computational Physics*, Vol. 228, No. 12, 2009, pp. 4588–4605.

[30] Liu, S. and Canfield, R. A., "Boundary velocity method for continuum shape sensitivity of nonlinear fluidstructure interaction problems," *Journal of Fluids and Structures*, Vol. 40, No. 0, 2013, pp. 284 – 301.

[31] Peskin, C. S., "Numerical analysis of blood flow in the heart," *Journal of computational physics*, Vol. 25, No. 3, 1977, pp. 220–252.

[32] Anderson, J. D. and Wendt, J., *Computational fluid dynamics*, Vol. 206, Springer, 1995.

[33] Saiki, E. and Biringen, S., "Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method," *Journal of Computational Physics*, Vol. 123, No. 2, 1996, pp. 450–465.

[34] Zhu, L. and Peskin, C. S., "Interaction of two flapping filaments in a flowing soap film," *Physics of Fluids (1994-present)*, Vol. 15, No. 7, 2003, pp. 1954–1960.

[35] Beyer, R. P. and LeVeque, R. J., "Analysis of a one-dimensional model for the immersed boundary method," *SIAM Journal on Numerical Analysis*, Vol. 29, No. 2, 1992, pp. 332–364.

[36] Peskin, C. S. and McQueen, D. M., "A three-dimensional computational method for blood flow in the heart I. Immersed elastic fibers in a viscous incompressible fluid," *Journal of Computational Physics*, Vol. 81, No. 2, 1989, pp. 372–405.

[37] Fauci, L. J. and Peskin, C. S., "A computational model of aquatic animal locomotion," *Journal of Computational Physics*, Vol. 77, No. 1, 1988, pp. 85–108.

[38] Kempe, T., Lennartz, M., Schwarz, S., and Fröhlich, J., "Imposing the free-slip condition with a continuous forcing immersed boundary method," *Journal of Computational Physics*, Vol. 282, 2015, pp. 183–209.

[39] Uhlmann, M., "An immersed boundary method with direct forcing for the simulation of particulate flows," *Journal of Computational Physics*, Vol. 209, No. 2, 2005, pp. 448–476.

[40] Mittal, R. and Iaccarino, G., "Immersed boundary methods," *Annu. Rev. Fluid Mech.*, Vol. 37, 2005, pp. 239–261.

[41] Goldstein, D., Handler, R., and Sirovich, L., "Modeling a no-slip flow boundary with an external force field," *Journal of Computational Physics*, Vol. 105, No. 2, 1993, pp. 354–366.

[42] Durlofsky, L. and Brady, J., "Analysis of the Brinkman equation as a model for flow in porous media," *Physics of Fluids*, Vol. 30, No. 11, 1987, pp. 3329–3341.

[43] Arquis, E. and Caltagirone, J., "Sur les conditions hydrodynamiques au voisinage dune interface milieu fluide-milieu poreux: applicationa la convection naturelle," *CR Acad. Sci. Paris II*, Vol. 299, 1984, pp. 1–4.

[44] Angot, P., "Analysis of singular perturbations on the Brinkman problem for fictitious domain models of viscous flows," *Mathematical methods in the applied sciences*, Vol. 22, No. 16, 1999, pp. 1395–1412.

[45] Gazzola, M., Chatelain, P., Van Rees, W. M., and Koumoutsakos, P., "Simulations of single and multiple swimmers with non-divergence free deforming geometries," *Journal of Computational Physics*, Vol. 230, No. 19, 2011, pp. 7093–7114.

[46] Kevlahan, N. K.-R. and Ghidaglia, J.-M., "Computation of turbulent flow past an array of cylinders using a spectral method with Brinkman penalization," *European Journal of Mechanics-B/Fluids*, Vol. 20, No. 3, 2001, pp. 333–350.

[47] Mohd-Yusof, J., "Combined immersed-boundary/B-spline methods for simulations of ow in complex geometries," *Annual Research Briefs. NASA Ames Research Center= Stanford University Center of Turbulence Research: Stanford*, 1997, pp. 317–327.

[48] Verzicco, R., Mohd-Yusof, J., Orlandi, P., and Haworth, D., "LES in complex geometries using boundary body forces," *Center for Turbulence Research Proceedings of the Summer Program, NASA Ames= Stanford University*, 1998, pp. 171–186.

[49] Verzicco, R., Fatica, M., Iaccarino, G., Moin, P., and Khalighi, B., "Large eddy simulation of a road vehicle with drag-reduction devices," *AIAA journal*, Vol. 40, No. 12, 2002, pp. 2447–2455.

[50] Iaccarino, G. and Verzicco, R., "Immersed boundary technique for turbulent flow simulations," *Applied Mechanics Reviews*, Vol. 56, No. 3, 2003, pp. 331–347.

[51] Clarke, D. K., Hassan, H., and Salas, M., "Euler calculations for multielement airfoils using Cartesian grids," *AIAA journal*, Vol. 24, No. 3, 1986, pp. 353–358.

[52] Kirkpatrick, M., Armfield, S., and Kent, J., "A representation of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid," *Journal of Computational Physics*, Vol. 184, No. 1, 2003, pp. 1–36.

[53] Hu, X., Khoo, B., Adams, N. A., and Huang, F., "A conservative interface method for compressible flows," *Journal of Computational Physics*, Vol. 219, No. 2, 2006, pp. 553–578.

[54] Udaykumar, H., Mittal, R., and Shyy, W., "Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids," *Journal of computational physics*, Vol. 153, No. 2, 1999, pp. 535–574.

[55] Borrvall, T. and Petersson, J., "Topology optimization of fluids in Stokes flow," *International journal for numerical methods in fluids*, Vol. 41, No. 1, 2003, pp. 77–107.

[56] Challis, V. J. and Guest, J. K., "Level set topology optimization of fluids in Stokes flow," *International journal for numerical methods in engineering*, Vol. 79, No. 10, 2009, pp. 1284–1308.

[57] Deaton, J. D. and Grandhi, R. V., "A survey of structural and multidisciplinary continuum topology optimization: post 2000," *Structural and Multidisciplinary Optimization*, Vol. 49, No. 1, 2014, pp. 1–38.

[58] LeVeque, R. J. and Li, Z., "Immersed interface methods for Stokes flow with elastic boundaries or surface tension," *SIAM Journal on Scientific Computing*, Vol. 18, No. 3, 1997, pp. 709–735.

[59] Pingen, G., Evgrafov, A., and Maute, K., "Topology optimization of flow domains using the lattice Boltzmann method," *Structural and Multidisciplinary Optimization*, Vol. 34, No. 6, 2007, pp. 507–524.

[60] Mase, G. T., Smelser, R. E., and Mase, G. E., *Continuum mechanics for engineers*, CRC press, 2009.

[61] Haftka, R. T. and Grandhi, R. V., "Structural shape optimizationa survey," *Computer Methods in Applied Mechanics and Engineering*, Vol. 57, No. 1, 1986, pp. 91–106.

[62] Gobal, K., Grandhi, R. V., and Kolonay, R. M., "Continuum Sensitivity Analysis for Structural Shape Design Variables Using Finite-Volume Method," *AIAA Journal*, Vol. 53, No. 2, 2014, pp. 347–355.

[63] Dowding, K. J. and Blackwell, B. F., "Sensitivity analysis for nonlinear heat conduction," *Journal of Heat Transfer*, Vol. 123, No. 1, 2001, pp. 1–10.

[64] Szopa, R., Siedlecki, J., and Wojciechowska, W., "Second order sensitivity analysis of heat conduction problems," *Scientific Research of the Institute of Mathematics and Computer Science*, Vol. 4, No. 1, 2005, pp. 255–263.

[65] Sorli, K. and Skaar, I. M., "Sensitivity Analysis For Thermal Design And Monitoring Problems Of Refractories," Begel House Inc., April 19-24, 2004, Norway.

[66] Szabo, B. A. and Babuška, I., *Finite element analysis*, John Wiley & Sons, 1991.

[67] Wickert, D. P., "Least-squares, continuous sensitivity analysis for nonlinear fluid-structure interaction," Tech. rep., DTIC Document, 2009.

[68] Deaton, J. D. and Grandhi, R. V., "Stiffening of restrained thermal structures via topology optimization," *Structural and Multidisciplinary Optimization*, Vol. 48, No. 4, 2013, pp. 731–745.

[69] Jain, A., Jones, N. P., and Scanlan, R. H., "Coupled flutter and buffeting analysis of long-span bridges," *Journal of Structural Engineering*, 1996.

[70] Arrigan, J., Pakrashi, V., Basu, B., and Nagarajaiah, S., "Control of flapwise vibrations in wind turbine blades using semi-active tuned mass dampers," *Structural Control and Health Monitoring*, Vol. 18, No. 8, 2011, pp. 840–851.

[71] Farhat, C., Van der Zee, K. G., and Geuzaine, P., "Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity," *Computer methods in applied mechanics and engineering*, Vol. 195, No. 17, 2006, pp. 1973–2001.

[72] Sotiropoulos, F. and Borazjani, I., "A review of state-of-the-art numerical methods for simulating flow through mechanical heart valves," *Medical & biological engineering & computing*, Vol. 47, No. 3, 2009, pp. 245–256.

[73] Kern, S. and Koumoutsakos, P., "Simulations of optimized anguilliform swimming," *Journal of Experimental Biology*, Vol. 209, No. 24, 2006, pp. 4841–4857.

[74] Lomtev, I., Kirby, R., and Karniadakis, G., "A discontinuous Galerkin ALE method for compressible viscous flows in moving domains," *Journal of Computational Physics*, Vol. 155, No. 1, 1999, pp. 128–159.

[75] Farhat, C., "CFD-based nonlinear computational aeroelasticity," *Encyclopedia of computational mechanics*, Vol. 3, 2004, pp. 459–480.

[76] Cheng, Y., Oertel, H., and Schenkel, T., "Fluid-structure coupled CFD simulation of the left ventricular flow during filling phase," *Annals of biomedical engineering*, Vol. 33, No. 5, 2005, pp. 567–576.

[77] Majumdar, S., Iaccarino, G., and Durbin, P., "RANS solvers with adaptive structured boundary non-conforming grids," *Annual Research Briefs, Center for Turbulence Research, Stanford University*, 2001, pp. 353–466.

[78] Peskin, C. S., "Flow patterns around heart valves: a numerical method," *Journal of computational physics*, Vol. 10, No. 2, 1972, pp. 252–271.

[79] Lee, C., "Stability characteristics of the virtual boundary method in three-dimensional applications," *Journal of Computational Physics*, Vol. 184, No. 2, 2003, pp. 559–591.

[80] Shin, S. J., Huang, W.-X., and Sung, H. J., "Assessment of regularized delta functions and feedback forcing schemes for an immersed boundary method," *International Journal for Numerical Methods in Fluids*, Vol. 58, No. 3, 2008, pp. 263–286.

[81] Roma, A. M., Peskin, C. S., and Berger, M. J., "An adaptive version of the immersed boundary method," *Journal of computational physics*, Vol. 153, No. 2, 1999, pp. 509–534.

[82] Peskin, C. S., "The immersed boundary method," *Acta numerica*, Vol. 11, 2002, pp. 479–517.

[83] Kreissl, S., Pingen, G., and Maute, K., "An explicit level set approach for generalized shape optimization of fluids with the lattice Boltzmann method," *International Journal for Numerical Methods in Fluids*, Vol. 65, No. 5, 2011, pp. 496–519.