Shape Sensitivity Analysis for Coupled Fluid-Solid Interaction Problems

Koorosh Gobal

December 20, 2015

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Literature Review . . . 1
  - 1.2.1 Sensitivity Analysis . . . 1
  - 1.2.2 Immersed Boundary Method . . . 5
  - 1.2.3 Sensitivity analysis for IB method . . . 7
- 1.3 Research Contribution . . . 8

2 Design Sensitivity Analysis **9**
- 2.1 General Formulation . . . 9
- 2.2 Benchmark Case . . . 10
- 2.3 Discrete Sensitivity Formulation . . . 11
- 2.4 Continuum Sensitivity Formulation . . . 13
- 2.5 Summary . . . 18

3 The Immersed Boundary Method **19**
- 3.1 Introduction . . . 19
- 3.2 Governing Equations . . . 20
- 3.3 Benchmark Case . . . 21
- 3.4 Immersed Boundary Classification . . . 22
- 3.5 Discrete Forcing Method . . . 22
  - 3.5.1 Formulation . . . 22
  - 3.5.2 Implimentation for Couette Flow Problem . . . 22
- 3.6 Continuum Forcing Method . . . 22
  - 3.6.1 Classical IB method . . . 24
  - 3.6.2 Virtual boundary method . . . 26
  - 3.6.3 Penalization method . . . 27

List of Figures

1.1 Sensitivity calculation techniques. 2
1.2 Example of conforming and nonconforming meshes. 5
1.3 Modified mesh near the solid boundary for cut cell method. 7

2.1 One dimensional domain with heat conduction. 10
2.2 One dimensional computational domain for heat conduction. 11
2.3 Comparison between the analytical and finite difference solution for 1D heat equation. 12
2.4 Sensitivity analysis verification. 13
2.5 Comparison between the analytical and continuum sensitivity analysis2 solution for 1D heat equation. 18

3.1 1D benchmark case for IB method. 22
3.2 Comparison between different formulations for ϕ 24
3.3 Comparison between IB and analytical results for different locations of stationary wall. 26
3.4 Comparison between IB and analytical results for different locations of stationary wall. 29
3.5 Flow through a porous pipe. 30

List of Tables

2.1 Comparison between the governing and sensitivity equations. 17

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 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 multidisciplinary nature. For most FSI problems, analytical solutions to are not available, and physical experiments are limited in scope. Therefore, to get more insight in the physics involved in the complex interaction between fluids and solids, numerical simulations are used. The numerical solutions are conducted based on Computational Fluid Dynamics (CFD) models for the flow 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. The effort and time required to take a geometry from a CAD package, clean up the model, and generate a mesh is usually a large portion of the overall human time required for the simulation. This cannot be automated for complex and moving domains. The Immersed Boundary (IB) method, reduces the amount of time needed for the fluid flow simulations and provides fast results by directly addressing the challenges associated with this issue.

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 yet implemented in commercial CFD packages. Therefore, most gradient optimization techniques relay on finite difference method for sensitivity calculation when solving FSI problem that are prone to errors.

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 solvers as black-box. This means that we can solve both the governing equations and 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 details in the following Chapters.

1.2 Literature Review

1.2.1 Sensitivity Analysis

Sensitivity analysis consists of computing the derivatives of solution of the governing equations, i.e. displacement, velocity, or pressure, with respect to one of several independent design

variables, i.e. shape of boundaries or size of elements. There various applications for sensitivity information such as improving the accuracy of surrogate models as in gradient enhanced Kriging [1] or uncertainty quantification [2]. 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, using 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 methods: i) numerical, ii) analytical, and iii) automatic differentiation.

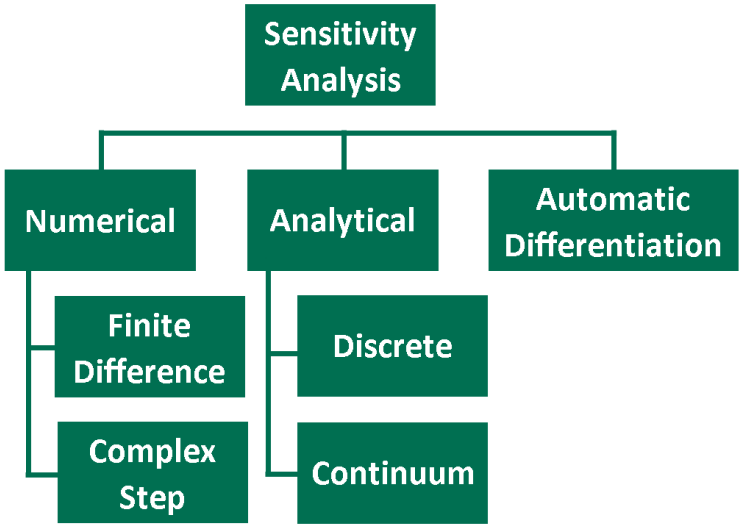


Figure 1.1: Sensitivity calculation techniques.

The Finite Difference (FD) method is probably the easiest method to implement for calculating the sensitivity of a variable. The fact that they can be implemented even when a given computational model is a black box makes most gradient based optimzation algorithms perform finite differences by default when the used does not provide the required gradients. However, the computational cost associated with finite difference for large systems can become very large. For a system with n number of design variables, the analysis needs to be done $n+1$ time to calculate the design sensitivities. Furthermore, to ensure the accuracy, convergence study needs to be done for selecting the appropriate step size for finite difference. The inaccuracy of finite differencing could result in convergence difficulties and inaccurate optimum results. On the other hand, Complex Step (CS) method avoids the loss of precision in finite differences approximation of sensitivities by employing complex arithmetic [3]. The complex step derivative is calculated as shown in Equation (1.1).

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

This means that we perturb the design variable by an imaginary value of ih and then look at the imaginary portion of the resulting response. Using the complex step method, we can choose a small step size for h without loosing accuracy. 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 to computer programs [4]. In the AD approach, the chain rule of differentiation is applied to every line in the program. This assumes 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 many 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. However, in all of these works, it is needed to have access to the source code and modify the solver extensively to calculate the sensitivities. This make the use of this method for general purpose codes infeasible since the source code is usually not available.

The short comings of numerical and AD techniques, fuels the research for more sophisticated methods for sensitivity calculation which 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 derived and solved. Typically the continuum equations 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 [5]. This system of equations is later solved for analytical sensitivities.

The discrete sensitivity analysis has been historically the method of choice to calculate the high-accurate sensitivity when the details of analysis are available [6]. 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 [7]. They used Euler flow as the aerodynamics theory where they optimized different configurations for transonic and supersonic regimes. However, they only focused on the flow. 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 shape (OML) of the aircraft. In their work, the discretization details of the solver needs to be known to calculate the sensitivities. As a matter of fact, source code modification is essential in all the other related research in the area as well [8, 9, 10].

As mentioned before, sensitivity calculation is the essential part of gradient based optimization. However, many general purpose solvers do not have this capability. This is even more prominent in the fluid mechanics community where none of the available commercial software packages such as ANASYS Fluent or CFX has analytical sensitivity analysis capabilities. In optimization loop, they relay on finite difference as a method for sensitivity calculation. As mentioned in the previous paragraph, for the discrete sensitivity methods, intimate knowledge and access to the source codes of these software is needed to calculate the sensitivities. 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 operate on the governing equations before they are discretized. These method are commonly known as Continuum Sensitivity Methods [11].

The Continuum Sensitivity Analysis (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 difference 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 vector valued functions. I 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 year, the computational fluids community adopted the local CSEs, since its formulation is consistent with the governing equations [12, 13]. 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 to each other. This will be explained in more details in next chapter.

Overally, the sensitivity analysis in general is more matured is the concept of structural optimization than for problems of fluid dynamics. Nevertheless, neither of these disciplines do not typically employ CSA for the sensitivity calculation. Aurora and Haug [14], followed by Dems and Mroz [15], 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 the 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.

The first application of CSA in optimization problems for fluid dynamics is the work by Borggaard and Burns [16] for shape sensitivity analysis of inviscid supersonic flows over rigid bodies. Stanley and Stewart [17] 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 [18] 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 [19]. 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.

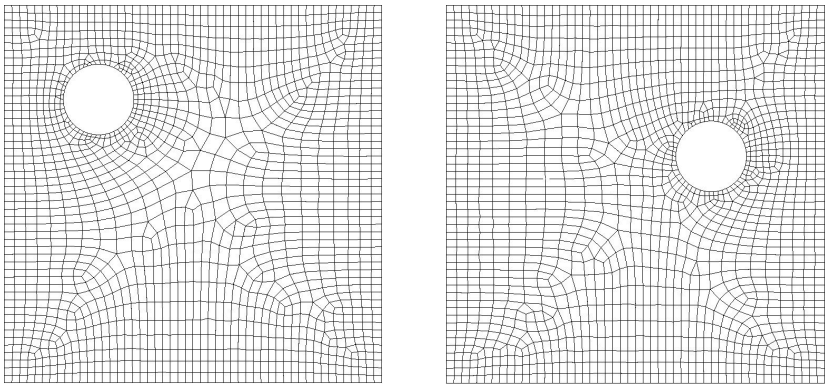
In almost all of the research done on sensitivity calculation of flow field to shape design variable, body conformal grids were used. The conforming mesh methods consider the interface conditions as physical boundary conditions, which treat the interface location as part of the solution and requires 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 [20]. 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 [21]. This adds to the computational cost of 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 method that does not require fluid domain mesh modification for the optimization iterations. On 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 simulation methods for flow over complex bodies are based on a body-fitted multi-block or unstructured grid methods. However, in the last decade another class of techniques, called immersed boundary methods, have been introduced to model the flow around solid boundaries. The term immersed boundary is first used by Peskin [22] to simulate the blood flow through heart valves. What distinguishes this method from the other methods of representing the solid boundaries is that the flow is solved on the Cartesian grid that does not conform to the solid 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 solution of the 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 [23]. If a finite-volume technique is employed, then the integral form of the governing equations is discretized and the geometrical information regarding the grid is incorporated directly into the discretization. If an unstructured grid is employed, then either a finite-volume or a finite-element methodology can be used.



(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 four categories: i) Continuous forcing, ii) discrete forcing, iii) penalization, and iv) cut-cell method.

Continuous forcing method

This is the originaly used by Peskin [22] and later developed by others researchers [24, 25, 26]. 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 buy 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 resets in representing the Dirac delta function as a discrete function that has the same properties. This method is applied to variety of problems, including Cardiac blood flow [27], animal locomotion [28], multiphase flows [29], and particle sedimentation [30].

Peskin's method is well suited for the elstic boundaries. For stiff boundaries, the constitutive laws of the solid boundaries will result in instabilities in the solution of governing equations [31]. The virtual boundary method of Goldstein [32] 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.

Discrete forcing method

This discrete formulation of IB method was first introduced by Mohd-Yusof [33] 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 [33] and Verzicco et al. [34] this is done by modifying the discretization stencil in the vicinity of the boundary curve so that the velocity values on this boundary satisfies a pre-defined condition. The major advantage of this approach is the lack of user specified parameters however, its implementation depends strongly on the discretization approach used for the analysis. Therefore, it cannot easily implemented in commercial codes. This implementation of discrete forcing method has be applied to many different problems such as turbulent flow inside an internal combustion engine [34], flow past 3D bluff bodies [35], and cylindrical stirred tank [36].

Penalization method

The penalization method is based on the Brinkman equation that describes the flow of a fluid 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 [37]. This approach was first proposed by Arquis and Caltagirone where they imposed the boundary conditions by adding

the penalization terms to the momentum equations [38]. 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 be argued that the solution of the penalized incompressible NS equations converges to the exact solution as the porosity approaches zero [39] 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 [40, 41].

Cut-cell method

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

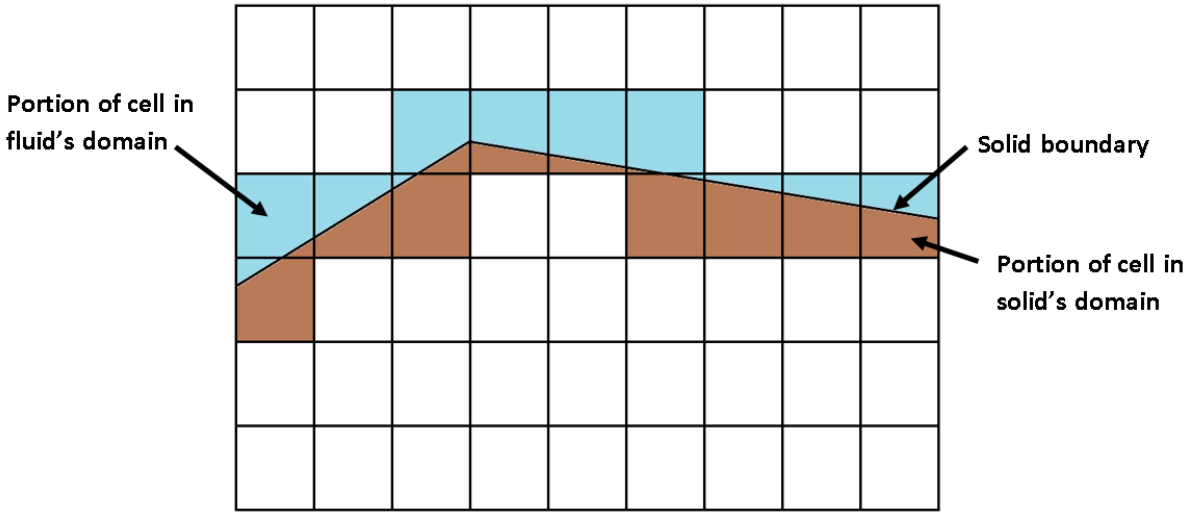


Figure 1.3: Modified mesh near the solid boundary for cut cell method.

The cut-cell method was first introduced Clarke [42] for inviscid flows. This method have been applied to both collocated and staggered grids [43] however, most of the applications were focused on 2D flows [44, 45]. 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 and resolving the stability issues. However, there has not been much effort in the sensitivity calculation of a IB formulation. Most of the sensitivity analysis research is developed for the penalization framework. Borrvall and Petersson applied the penalization method for topology optimization of fluids in Stokes flow [46]. 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. They also had constraint on the volume of domain that can be penalized. Challis and Guest investigated the level set formulation for the topology optimization of fluids in Stokes flow [47]. They used the penalization technique to formulate their problem where they were able to accurately model 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 pipe in 3D.

The penalization method is probably the easiest of the IB methods to implement. This is because it does not have a free parameter such a 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 domain are differentiated based a scalar function (porosity) which is very similar to the density based approaches in the topology optimization community [48]. 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 reused [49, 50]. Similar to topology optimization, the shortcoming of penalization technique is in its accuracy for representing the boundaries. Since the nodes are assigned 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

Due to the shortcomings of the penalization method, in this research we develop a senstivitiy analysis based on continuous forcing IB method. Since the sensitivity analysis is done using the continuum sensitivity analysis, only continuum formulations of IB are possible in this research. The continuum sensitivity formulation enables us to reuse the differential operators that we already developed for solving the governing equations. The IB method, removes the need to mesh deformation so that same computational mesh can be reused throughout the analysis. Moreover, since the mesh nodes do not move, the local and total form of the sensitivities are the same. Therefore, no extra computation is needed to converting the local to total form of the sensitivities.

Chapter 2

Design Sensitivity Analysis

In this chapter, the concept behind discrete and continuum sensitivity formulation is discussed and the general approach for deriving the sensitivity equations is presented. 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. The difference between local and total formulation of the sensitivity response is discussed and finally the independence of continuum sensitivity formulation to discretization method is proven. This enables use to reuse the solver of governing equation to calculate the sensitivity response. This is typically not possible for discrete sensitivity formulation.

2.1 General Formulation

The general computational domain is defined as shown in Figure **??**. The response variable on this domain can be from the fluid, i.e. pressure or velocity, or the solid, i.e. displacements. Nevertheless, the response is calculated using a governing equation subject to boundary conditions. In this work, the governing equation and boundary conditions are represented in the functional form as shown in Equation (2.1).

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

where $\mathbf{A}$ is the governing equation such as Navier-Stokes equations or elastic equations and $\mathbf{B}$ is the boundary condition definition. $\mathbf{u}$ 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 $\mathbf{g}$ is the value of governing equation and boundary condition. The sensitivity of response variable with respect to i-th design variable b_i, $\partial\mathbf{u}/\partial b_i$, can be calculate using some sensitivity analysis technique. This is done in the following sections.

The total sensitivity of response variable, $\mathbf{u}$ with respect to the i-th design variable is written as

$$\frac{D\mathbf{u}}{Db_i} = \frac{\partial\mathbf{u}}{\partial b_i} + \frac{\partial\mathbf{u}}{\partial\mathbf{x}} \cdot \frac{\partial\mathbf{x}}{\partial b_i} \tag{2.2}$$

The total derivative is known as material derivative in continuum mechanics [51]. This total sensitivity defines the change of response variable, $\mathbf{u}$, subjected 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 this point in space due to change in the design variable. This is specially applicable to shape sensitivity calculating where the change in design variable, will cause the material points to move [52]. The convective term consists of two separate gradients: i) $\partial \mathbf{u}/\partial \mathbf{x}$ which represents the spatial gradient of the response variable in the domain, and ii) $\partial \mathbf{x}/\partial b_i$ which defines the sensitivity of the spatial domain with respect to the change in design variable. The later defines how the computational nodes moves as the design variable changes. The response variable spatial gradient term can be calculated using the analysis results, using finite difference approach or derivative of shape functions in FEA formulation. Calculation of domain sensitivity, $\partial \mathbf{x}/\partial b_i$, requires more attention.

A common approach to calculate the domain sensitivity, is to use the same techniques used to deform the body-conformal mesh in a CFD simulation. These methods are usually based on representing the computational grid as a system of springs that connected to each other at the nodes. This system can be modeled and solved using structural analysis techniques, where the sensitivities can be easily added to its formulation. This is effectively a shape sensitivity analysis for the structural analyses [53]. This step can be removed from the analysis if the computational domain does not affected by design variable since $\partial x/\partial b$ is equal to zero for this case. This is removes the extra step of doing the structural shape sensitivity analysis for the mesh and also response variable gradient calculation. As mentioned in Chapter 1, by using the immersed boundary method the mesh definition is decoupled from the boundary shape. Therefore, domain sensitivity is equal to zero [54]. 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 **??**.

2.2 Benchmark Case

To compare the discrete and continuum sensitivity analysis we chose the one-dimensional heat transfer analysis in a rod. The temperature in governing by the Laplace equation as shown in Equation (2.3).

$$\frac{\partial^2 T}{\partial x^2} = 0 \tag{2.3}$$

where T is the temperature and x is spatial variable. The boundary conditions are defined as constant temperatures at the two ends of the domain as shown in Figure 2.1. The length of domain is selected as L.

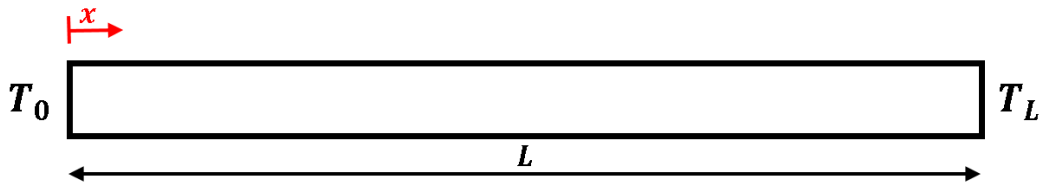


Figure 2.1: One dimensional domain with heat conduction.

The analytical solution of Equation (2.3) can be written as follows

$$T = \frac{T_L - T_0}{L} x + T_0 \tag{2.4}$$

The analytical sensitivity of the temperature with respect to beam's length can be calculated by differentiating Equation (2.4) with respect to L.

$$\frac{\partial T}{\partial L} = -\frac{T_L - T_0}{L^2} x \tag{2.5}$$

Since the analytical derivatives are known, we can compare it to the discrete and continuous sensitivity results.

2.3 Discrete Sensitivity Formulation

To formulate the discrete sensitivity equation, we start by discretizing the governing equation (2.3) using finite difference method. It should be noted that the finite difference is used for discretization of continuum governing equation and not sensitivity calculation. The design variable effects the shape of the domain which is the distance between the nodes in the discrete manner. Therefore, for the sake of sensitivity analysis it is required to keep the nodal distances in the discretized solution as well. We use 6 nodes to discretize the domain as shown in Figure 2.2.

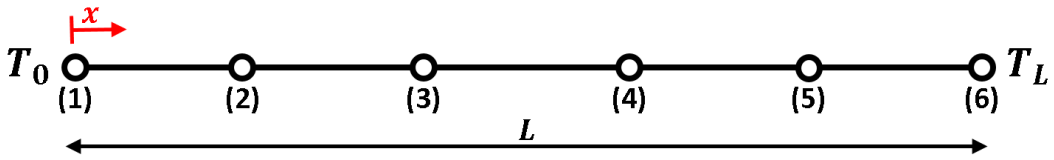


Figure 2.2: One dimensional computational domain for heat conduction.

To discretize Equation (2.3), the first step is to approximate the second derivative. This is done by writing the Taylor series expansion at at arbitrary location x_i.

$$T(x) = T(x_0) + \left.\frac{\partial T}{\partial x}\right|_{x_i} (x - x_i) + \left.\frac{\partial^2 T}{\partial x^2}\right|_{x_i} (x - x_i)^2 + \mathcal{H}.\mathcal{O}.\mathcal{T}$$

To approximate the second derivative at the arbitrary node x_i using central method we need to use the neighbouring nodes. The second order approximation for the second order derivative of central difference method is shown in Equation (2.6). The discretized equations are differentiated with respect to shape design variable. This requires the nodal distances to be included in the discretized equations. 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}.

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1}\Delta_{iL} - T_i(\Delta_{iL} + \Delta_{iR}) + T_{i+1}\Delta_{iR}}{\frac{1}{2}\left[\Delta_{iL}\Delta_{iR}^2 + \Delta_{iL}^2\Delta_{iR}\right]} \tag{2.6}$$

The approximation of the second derivative in Equation (2.3) is done using definitions in Equation (2.6) for each node excluding the boundary nodes. As mentioned in the previous section, the nodal values at boundary nodes, (1) and (2) are known. The discretized governing equations is written in the matrix form as shown in Equation (2.7).

$$\begin{bmatrix} \frac{-2}{\Delta_1\Delta_2} & \frac{2}{\Delta_1\Delta_2+\Delta_1^2} & 0 & 0 \\ \frac{2}{\Delta_3^2+\Delta_2\Delta_3} & \frac{-2}{\Delta_2\Delta_3} & \frac{2}{\Delta_2\Delta_3+\Delta_2^2} & 0 \\ 0 & \frac{2}{\Delta_4^2+\Delta_3\Delta_4} & \frac{-2}{\Delta_3\Delta_4} & \frac{2}{\Delta_3\Delta_4+\Delta_3^2} \\ 0 & 0 & \frac{2}{\Delta_5^2+\Delta_4\Delta_5} & \frac{-2}{\Delta_4\Delta_5} \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = - \begin{bmatrix} \frac{2T_1}{\Delta_2^2+\Delta_1\Delta_2} \\ 0 \\ 0 \\ \frac{2T_6}{\Delta_5\Delta_4+\Delta_5^2} \end{bmatrix} \tag{2.7}$$

To verify the discretization process, we compare the analytical solution of this problem with the result of Equation (2.7) in Figure 2.5. For this problem we choose $T_1 = 0$, $T_6 = 1$, and $L = 1$. As shown in this figure, the discrete and continuum results match very well.

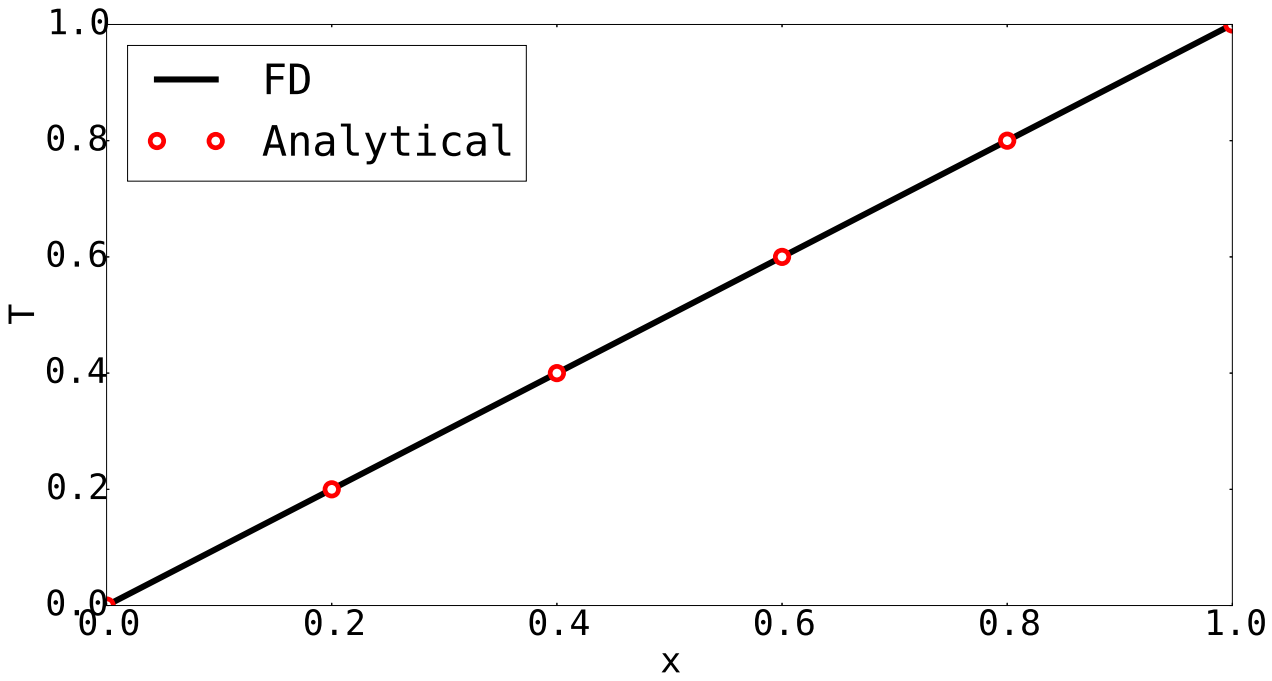


Figure 2.3: Comparison between the analytical and finite difference solution for 1D heat equation.

The sensitivity equations are derived by differentiating the discretized governing equation of (2.7) with respect 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 change in shape not movement of material nodes.

To calculate the sensitivities, It is required to calculate the derivative of nodal distances in Equation (2.7) with respect to L. For a equally space grid, the nodal distance is written as

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

where L is the length of the domain, and n is the number of nodes chosen to discretize the domain. The sensitivity of nodal distances to the length of the domain is calculated as

$$\frac{\partial \Delta x}{\partial L} = \frac{1}{n-1} \tag{2.8}$$

The discretized equation is differentiated as shown in Equation (2.9). 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.

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} T_2' \\ T_3' \\ T_4' \\ T_5' \end{bmatrix} = \frac{1}{2}\frac{\partial \Delta}{\partial L}\frac{1}{\Delta} \begin{bmatrix} 0 \\ 0 \\ 0 \\ T_6 \end{bmatrix} - \underbrace{\frac{1}{2}\frac{\partial \Delta}{\partial L}\frac{1}{\Delta} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}}_{\text{effect of shape change}} \tag{2.9}$$

In order to better study above we mention the general sensitivity equation in the discrete form. Assume the the continuum governing equation, can be written in the discrete form as

$$[K]\,[U] = [F]$$

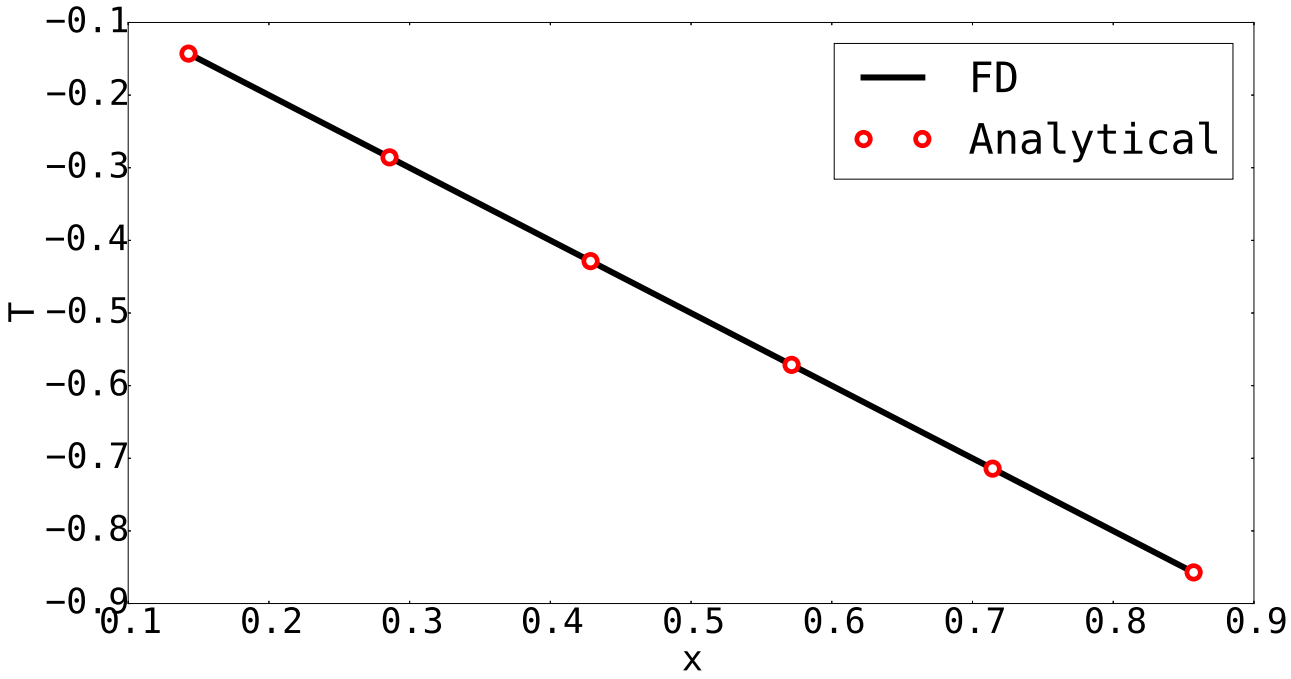


Figure 2.4: Sensitivity analysis verification.

where $[K]$ is the discrete operator that represents the governing equation, $[U]$ is the vector of response variable defined at each of the degrees of freedom, and $[F]$ is the vector of the boundary conditions. The sensitivity of this system with respect to design variable b is derived as

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

By comparing above equation with Equation (2.9), we can see that the last term represents the change in the stiffness matrix. This depends on how the governing equations are discretized. Therefore, to do the sensitivity analysis it is required to know how this matrix is put together so it can be differentiated. This is not possible in most cases since the details of discretization is not known for the commertial software packages.

To verify the results of Equation (2.9), we compared it with the analytical sensitivity results are shown in Figure 2.4.

2.4 Continuum Sensitivity Formulation

For a continuum system, the governing equations and boundary conditions can be written as

$$A(u,t;b) = 0 \quad \text{on } \Omega \tag{2.10a}$$
$$B(u,t;b) = g(x,t;b) \quad \text{on } \Gamma \tag{2.10b}$$
$$\tag{2.10c}$$

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 sensitivity of response, it is required to calculate the total derivative of Equation (2.10). This is broken into calculating the local sensitivity of the system of governing equation followed by converting the local sensitivities to their total form using Equation (2.2).

The governing equation of (2.10) is differentiated as follows

$$A(u', t; b) + A'(u, t; b) = 0 \quad \text{on } \Omega \tag{2.11a}$$
$$B(\dot{u}, t; b) + \dot{B}(u, t; b) = \dot{g}(x, t; b) \quad \text{on } \Gamma \tag{2.11b}$$
$$\tag{2.11c}$$

where u' and $\dot{u}$ 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 nonintrusively. However, in order to capture the effect of shape change, the boundary conditions need to be differentiated in total form. The boundary condition is further simplified by assuming linearly. This is a valid assumption for many practical cases such structural analysis or computational fluid dynamics. The boundary conditions are usually in the form of known gradient or value, i.e. outflow and free-slip wall for CFD and predefined displacement for structural analysis, at the boundary. It should be noted that this assumption is not valid for problems such as contacts. The boundary condition is written as

$$\begin{aligned} &B(\dot{u}, t; b) = \dot{g}(x, t; b) \Rightarrow \\ &B(u', t; b) = \dot{g}(x, t; b) - B(\frac{\partial u}{\partial x} \cdot \frac{\partial x}{\partial b}, t; b) \end{aligned} \tag{2.13}$$

To study the differentiated governing equation of Equation (2.11), we will assume two situations: i) linear analysis, ii) nonlinear analysis.

For the linear analysis, the governing of (2.11) is written as

$$A(u', t; b) = 0 \tag{2.14}$$

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

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.15}$$

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 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

$$\frac{\partial^2}{\partial x^2}\left(\frac{\partial T}{\partial b}\right) = \frac{1}{\alpha}\frac{\partial}{\partial t}\left(\frac{\partial T}{\partial b}\right) \tag{2.16}$$

By comparing Equations (2.15) and (2.16) we can see that the differential operators, $\partial^2/\partial x^2$, and $\partial/\partial t$ remain unchanged. Therefore, same analysis can be used for solving this system of governing equations for the sensitivity variable, $\partial T/\partial b$.

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

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} = 0 \tag{2.17a}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} = 0 \tag{2.17b}$$

where u and v are the velocity components in x and y directions respectively. p is the pressure, t is time, x, and y are spatial coordinates. We can rewrite Equation (2.17) in terms of differential operators too.

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

where $\mathcal{T}$ is the time derivative operator ($\partial/\partial t$), $\mathcal{C}$, is the convective operator ($u\partial/\partial x + v\partial/\partial y$). $\mathcal{G}_x$ 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 it should be differentiated alongside response variables as well. As a result, the sensitivity equations can be written in the operator form as

$$\mathcal{T}u' + \mathcal{C}'u + \mathcal{C}u' + \mathcal{G}_x p' = 0 \tag{2.19a}$$

$$\mathcal{T}v' + \mathcal{C}'v + \mathcal{C}v' + \mathcal{G}_y p' = 0 \tag{2.19b}$$

where

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

Several interesting properties of CSA can be explained using Equation (2.19). First of all, although the original Euler equation is nonlinear due to 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 in terms of algorithms and simulation time compared to the original equations. The challenging expression that needs to be calculated in Equation (2.19) is the convective term.

The first step in solving the sensitivity equations is to get 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 as 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 above equation, $\bar{\mathcal{C}}$ in the convective operator evaluated using $\bar{u}$. It should be noted that in order 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 and the black-box solver will generate it for us. This input is 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.

As for the discrete sensitivity case, we use the continuum sensitivity formulation for calculating the sensitivity of temperature with respect to length of a 1D domain. The domain is defined in Figure 2.1. The temperature in the domain is governed by Equation (2.3). The boundary conditions are defined as T_0 at $x = 0$ and T_L at $x = L$.

To get the governing equation for the sensitivity analysis, we need to differentiate the governing equations in the local form and boundary conditions in total form. The boundary conditions needs to be differentiated in the total form to reflect the movement of the boundary in space. The governing equations can be differentiated in the local form to calculate the local sensitivities. This is done in the following equation.

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

Since the differential operator does not depend on the design variable b (here the length of the domain, L) we can change the order of differentiation. This will give us the following equation for the sensitivity calculation. The next step would be calculating the required boundary conditions for this problem.

$$\frac{\partial^2}{\partial x^2}\left(\frac{\partial T}{\partial L}\right) = 0 \tag{2.20}$$

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

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

where $\mathcal{B}$ is the operator acting on the boundary. For this problem, it is equal 1. Above equation is differentiated in the total form since the boundaries are moving. This results in

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

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

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

To calculate the mesh sensitivity at the boundary, we need to know how the spatial variable x is related to the shape of the domain, L. For the 1D domain, the dependency of x and L 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.

$$\begin{cases} \dfrac{\partial T}{\partial b} = 0 & \text{at x = 0} \\ \dfrac{\partial T}{\partial b} = -\dfrac{\partial T}{\partial x} & \text{at x = L} \end{cases} \tag{2.21}$$

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

The governing equation of (2.20) with boundary condition (2.21) can be solved using the same solver that we used for solving the original governing equation since these two are effectively the same problem. The comparison between the original governing equation and the sensitivity equation is shown in Table 2.1.

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 2.1: Comparison between the governing and sensitivity equations.

The comparison between the solution of continuum sensitivity results and analytical solution for this problem is shown in Figure **??**. As shown here the two results match perfectly.

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.

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

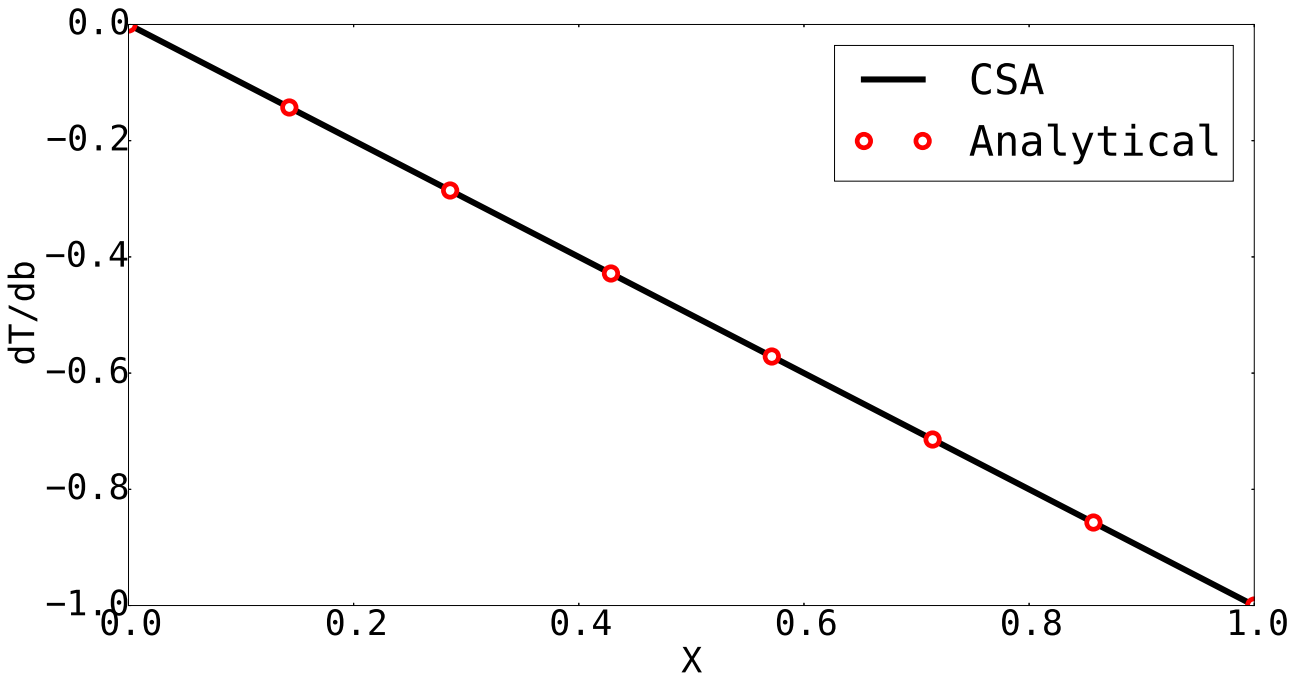


Figure 2.5: Comparison between the analytical and continuum sensitivity analysis2 solution for 1D heat equation.

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

2.5 Summary

In summery, 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 1D problem of heat transfer 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. We also showed that by using the continuum sensitivity analysis we can reuse the differential operators used in the solution of governing equations. 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 showed 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 with respect to design variables. So more details about the analysis needs to be known when using discrete approach compared to the continuum. Continuum sensitivity approach is used in this document 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 favourable 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 sensitivity calculation.

Chapter 3

The Immersed Boundary Method

Immersed boundary methods are class of techniques in computational fluid dynamics where the governing equations are solved on cartesian grid that does not conform to the shape of the body in the flow. This 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 special 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 chose 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 design requirements for different systems has been drastically changed compared to the initial designs where these computational methods have been applied. The 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, on way to reduce the infra-red signature of the aircraft engine is to remove it from hanging underneath the wing and put it inside of the aircraft. However, by doing this as massive heat source will be added the structure. The thermal expansion due to this excessive heat load needs to be incorporated into the structural analysis of the system. It requires a multidisciplinary analysis combining thermal analysis for heat transfer and structural analysis for thermal expansion and other structural loads [55].

The multidisciplinary analysis is required for many engineering application however the one that is used most is the interaction of fluid and a deformable structure. This is commonly known as a Fluid-Solid Interaction (FSI) problem. Fluidstructure interaction (FSI) problems are dealt with in many different engineering applications, such as fluttering and buffeting of bridges [56], vibration of wind turbine blades [57], aeroelastic response of airplanes [58]. FSI problems are also seen in blood flows in arteries and artificial heart valves [59], flying and swimming [60]. The conventional approach for simulating such problems is the Arbitrary LagrangianEulerian (ALE) method. ALE methods are based on body- conforming grids to track the location of the fluidstructure 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 complicated 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. When using structured grid, extremely efficient computational methods can be utilized on to solve the governing equations. The fluidstructure 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 methods. 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. This 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 complicated 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 clear for flows with moving boundaries. The ALE approach requires generating a new mesh or deforming the old 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 the 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 [61, 62, 63], the IB method still 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 implimentation. 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 unstructured grid to represent the shape, it is usually required to use mappings

(Jacobian) to convert the physical coordinate the a computational coordinate. This will become problematic when we have skewed elements with cause the mapping 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 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 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 can be calculated by integrating the pressure from the fluid solid over the solid boundaries. This will be the boundary load used for solving Equation (3.2). When the governing equation for the solid region is solved, the displacement of the solid domain is known. This is fed into the CFD solver to update the solid boundary location for fluid domain. This will change the solution of fluid's solver resulting in different pressure distribution. This process is repeated until a convergence is satisfied. The convergence can be defined as change in the structure deflection for two subsequent steps for 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 very interesting and it is not simple to write a one-dimensional analogue of the IB to capture all of its features. However, for a simple one dimensional problem we can understand different formulations of IB method and later apply it to higher dimensions. Moreover, for the sake of sensitivity analysis better understanding of the problem can be achieved for a simplified model. 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 structures 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.

We can also assume that the boundary conditions are periodic in x. Next suppose that there is a horizontal plate running through the length of this channel at $y = y_0$ and moving with u_0 velocity in horizontal direction. We assume no-slip condition where the fluid and the plate meet. This 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 can drop all terms in the NS equations containing u_j velocity. Due to the continuity equation (3.1a), there is no variation in the 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) into Equation (3.3). It should be noted that the continuum equation (3.1a) is automatically satisfied.

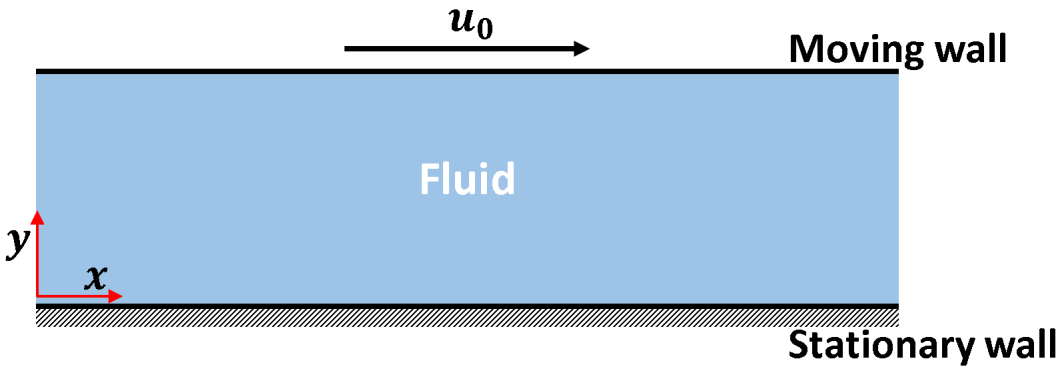


Figure 3.1: 1D benchmark case for IB method.

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

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

Equation (3.3) is transient in nature however, its steady state solution can be calculated by setting the time derivative equal to zero. This equation governs what is commonly known as Couette flow in introductory courses 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 result of the 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 their primary advantages and disadvantages. Moreover, each of the discussed IB techniques will be applied to the benchmark problem where the results are verified with the analytical solution. In general, the immersed boundary approach is based on modifying the NS equations for imposing the boundary conditions. This modification can be done in three different ways that leads to a fundamental dichotomy in IB method.

3.5 Discrete Forcing Method

3.5.1 Formulation

3.5.2 Implimentation for Couette Flow Problem

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 [**?**] 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

$$\frac{\partial X_k}{\partial t} = u(X_k, t) \tag{3.5}$$

where u is the velocity of the fluid at location X_k. The location of fluid nodes, x, does not necessary coincide with the location of the Lagrangian points. Thus, it is required to map the velocities from the Eulerian domain, where the fluid's equation of motion are solved, to Lagrangian nodes. In a purely 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 domain is shown in the following equation

$$\int_{\Omega_f} f(x)\delta(x - X_k) = f(X_k) \tag{3.6}$$

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. By defining the delta function at X_k and using velocity from CFD solver as function f, we can evaluate the needed velocity for IB at any arbitrary point X_k. Although this is the main idea of mapping data to Lagrangian domain, this approach becomes unstable in practice [**?**]. In the practical implementation of immersed boundary method, the effect of delta function need to be expanded to couple of nodes around X_k. This is achieved by relaxing the delta function. The relaxed delta function is generally refereed to as regularized delta function [**?**]. The regularized delta function is defined using Equation (3.7) in three dimensions.

$$\delta_h(x_1, x_2, x_3) = \frac{1}{dx_1 \cdot dx_2 \cdot dx_3}\phi\left(\frac{x_1 - \eta_1}{dx_1}\right)\phi\left(\frac{x_2 - \eta_2}{dx_2}\right)\phi\left(\frac{x_2 - \eta_3}{dx_3}\right) \tag{3.7}$$

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 (3.8). As shown here there are different ways of defining this function. The input of ϕ function is r which is defined as $(x_i - \eta_i)/dx_i$.

$$\phi(r) = \begin{cases} 1 - |r| & |r| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.8a}$$

$$\phi(r) = \begin{cases} \frac{1}{3}\left(1 + \sqrt{-3r^2 + 1}\right) & |r| \leq 0.5 \\ \frac{1}{6}\left(5 - 3|r| - \sqrt{-3(1 - |r|)^2 + 1}\right) & 0.5 \geq |r| \leq 1.5 \\ 0 & \text{otherwise} \end{cases} \tag{3.8b}$$

$$\phi(r) = \begin{cases} \frac{1}{8}\left(3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}\right) & 0 \leq |r| \leq 1 \\ \frac{1}{8}\left(5 - 2|r| + \sqrt{-7 + 12|r| - 4r^2}\right) & 1 \leq |r| \leq 2 \\ 0 & \text{otherwise} \end{cases} \tag{3.8c}$$

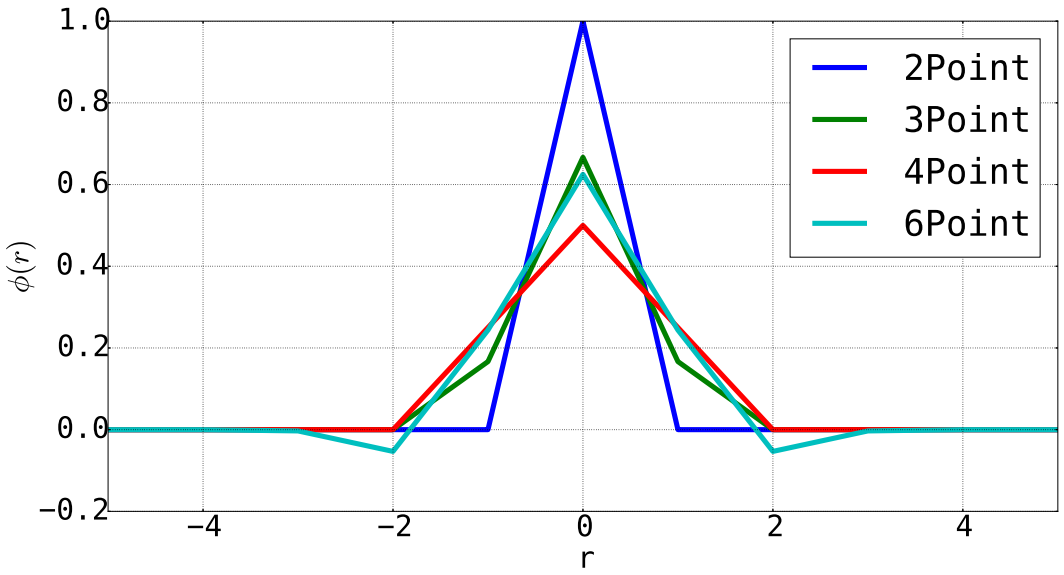


Figure 3.2: Comparison between different formulations for ϕ

$$\phi(r) = \begin{cases} \frac{61}{112} - \frac{11}{42}|r| - \frac{11}{56}|r|^2 + \frac{1}{12}|r|^3 + \frac{\sqrt{3}}{336}\left(243 + 1584|r| \right. & 0 \le |r| \le 1 \\ \left. -748|r|^2 - 1560|r|^3 + 500|r|^4 + 336|r|^5 - 112|r|^6\right)^{1/2} & \\ \frac{21}{16} + \frac{7}{12}|r| - \frac{7}{8}|r|^2 + \frac{1}{6}|r|^3 - \frac{3}{2}\phi\left(|r| - 1\right) & 1 \le |r| \le 2 \\ \frac{9}{8} - \frac{23}{12}|r| + \frac{3}{4}|r|^2 - \frac{1}{12}|r|^3 + \frac{1}{2}\phi\left(|r| - 2\right) & 2 \le |r| \le 3 \\ 0 & \text{otherwise} \end{cases} \tag{3.8d}$$

Equation (3.8a) is the 2-point delta function that does the linear interpolation between the points point [24]. Equation (3.8b) shoes the 3-point delta function used by Roma et al. [**?**], and Equation (3.8c) and (3.8d) are 4-point and 6-point delta functions used by Peskin [**?**]. The comparison between the shape of different $\phi(r)$ functions are shown in Figure **??**

Now we can write the mapping from the Eulerian nodes of the fluid solver to Lagrangian nodes as shown in Equation (3.9).

$$u(X_k) = \int_\Omega u(x)\delta(x - X_k)dx \tag{3.9}$$

where Ω is the computational domain, $u(x)$ is the velocity at the Eulerian nodes, x is the coordinate of the Eulerian nodes, $u(X_k)$ 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 upto this point. They diverge depending of 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 calculate using appropriate constitutive laws, i.e. Hooks law. This can be expressed as follows.

$$f(X_k) = \mathcal{M}(X_k) \tag{3.10}$$

where $\mathcal{M}$ is an operator which describes the properties of the boundaries. This force is calculated at the Lagrangian points and need 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 are 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

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} - f(t) \tag{3.11}$$

where $f(t)$ is the forcing function using to model the boundary. This equation can be discretized using the explicit backward Euler method as follows

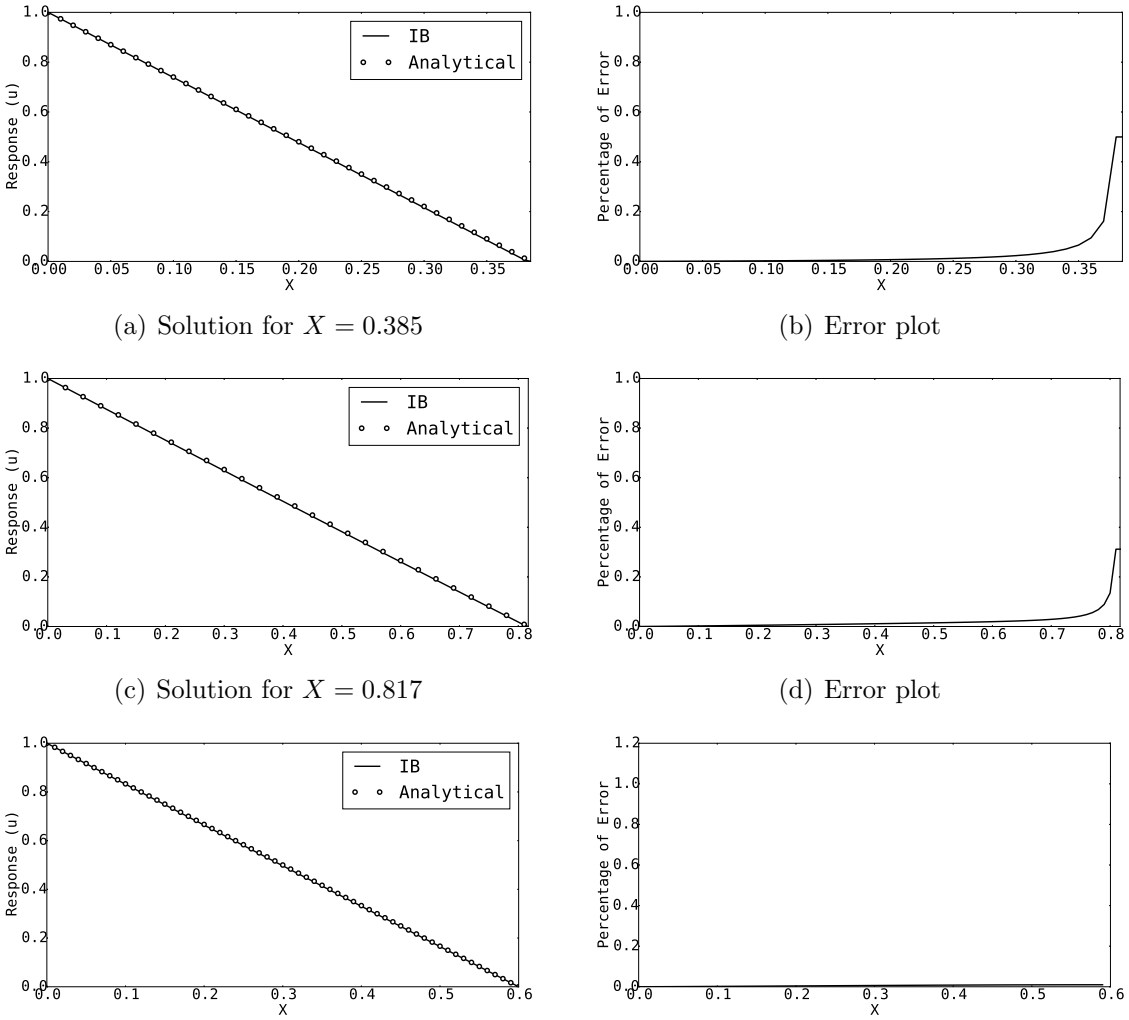
$$u^{n+1} = u^n + \Delta t \left(\frac{\partial^2 u^n}{\partial x^2} - f^n \right) \tag{3.12}$$

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 next time step, u^1, using Equation (3.12)
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: $F = K\left(X^{n+1} - X^0\right)$, 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 this problem we chose the time step as 0.1 and used 100 nodes to model the domain where the length of the domain is selected as 1.0. The velocity at which the wall is moving is selected as $1.0m/s$. We chose several different locations and different stiffness values for the wall. We verified the IB solution with the analytical results. We used a 2-point δ function to transfer data between Lagrangian and Eulerian domains. The results for this are shown in Figure 3.4.

As shown in Figure 3.4, we compared the solution of IB and analytical results using $X = 0.385$ and $X = 0.6$ for the position of the wall. $X = 0.385$ is a exactly in the middle of, $X = 0.817$ is somewhere between two nodes. $X = 0.6$ is exactly on one of the computational nodes. It can be seen that the IB method is able to capture the location of the wall within an acceptable accuracy. This error goes to zero when the location of Lagrangian point, i.e. is directly on top of the Eulerian node. It should be noted that this method is extremly sensitive to the value of K in defining the forcing terms whose selection is an iterative process. Choosing a large value for K will results in an unstable solution (divergence) whereas choosing a small value for K will reduce the accuracy of the method.



(a) Solution for $X = 0.385$

(b) Error plot

(c) Solution for $X = 0.817$

(d) Error plot

(e) Solution for $X = 0.817$

(f) Error plot

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

3.6.2 Virtual boundary method

The virtual boundary method is a different approach for imposing the effect of immersed boundary through force terms. This approach is well suited for both the rigid boundaries and also elastic boundaries. The difference between this approach and the classical IB method defined in the previous chapter is that it does not require the solution of the constitutive equations to calculate the force terms. This approach is based on the works of the works of Goldstein et al. [32] to simulate the flow around the rigid bodies. The forcing term in the virtual boundary method is defined as

$$F(X_k, t) = \alpha \int_0^t \left[u(X_k, t) - U(X_k, t)\right] dt + \beta \left[u(X_k, t) - U(X_k, t)\right] \tag{3.13}$$

where F is the required force at the k-th Lagrangian point, $u(X_k, t)$ is the velocity calculated from the 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 (3.13) 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 stand point, this equation represents a damped oscillator [36].

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.

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{1}{2}\left(\frac{\partial^2 u^{n+1}}{\partial x^2} + \frac{\partial^2 u^n}{\partial x^2}\right) \tag{3.14}$$

where n is the time step that we are at. 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 (3.12)
3. Map the velocity results to the Lagrangian nodes using the δ function
4. Based on the new velocity at $t = 1$ evaluate Equation (3.13).
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 less steps are required for this formulation. Moreover, there is no need to solve the constitutive equation for the solid domain which reduces the total computational cost.

For this problem, we are choosing the location of wall at two different locations, $x = 0.6$ and $x = 0.385$. The first node is exactly on top of a computational node and the second is between two nodes. We also looked at two different velocities for the moving wall, $u_0 = 1$ and $u_0 = 100$. The time step for solving Equation (3.14) is chosen as 0.1 and we use 100 nodes to model the domain where the length of the domain is selected as 1.0. We used 2-point and 4-point δ function to transfer data between Lagrangian and Eulerian domains. The results for these are shown in Figure **??**.

As shown in Figure 3.4, we compared the solution of IB and analytical results using $X = 0.385$ and $X = 0.6$ for the position of the wall. As can be seen here, the virtual boundary method results are in better agreement with the analytical results compated to the classical IB method. This is due to the fact the for IB method, the boundary can move and is not perfect. Therefore, the boundary conditions are assigned in an approximate manner. Moreover, as can be seen here, the virtual boundary method can handle different magnitude for the velocity without modification to the method. This makes this approach a more general approach for rigid boundaries compared to classical IB approach. Moreover, as can be seen in these results the location of boundary is captured well even if its location does not coincide the the computational nodes.

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 [38]. 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 is a fraction of the volume of voids over the total volume, between 0 and 1. For the solid domain the porosity value is near zero whereas for the fluid domain its value is close to one. 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 over a given distance.

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

$$Q = \frac{\kappa A \Delta p}{\mu L} \tag{3.15}$$

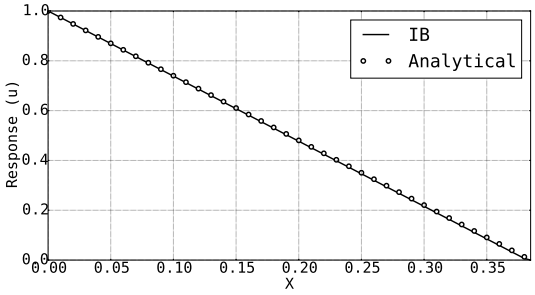
where Q is the total flow discharge, κ is the permeability of the domain, Δp is the pressure drop due to the porosity between two ends of the pipe, μ is the fluid's viscosity, and L is the length of the domain. Darcy's law can be considered as a relation between the flow velocity and pressure drop. This pressure drop is used in the penalization method to represent the solid boundaries by modelling the force term using Equation (3.15). The force term is defined as follows

$$f = -\mathcal{H}(\mathcal{X})\frac{\mu}{\kappa}v \tag{3.16}$$

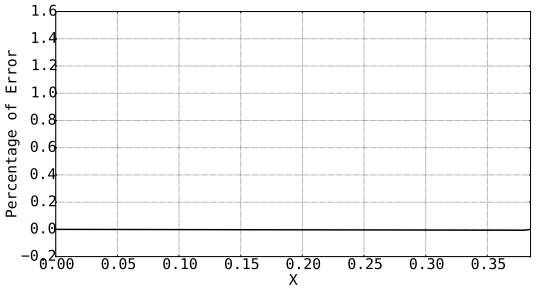
where v is is the velocity of the flow. In order to apply the force term only to the region inside the the solid boundary, the penalization force is multiplied by a Heaviside function, $\mathcal{H}(\mathcal{X})$, that is a function of relative distance of the points in the domain to the boundary of the 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.

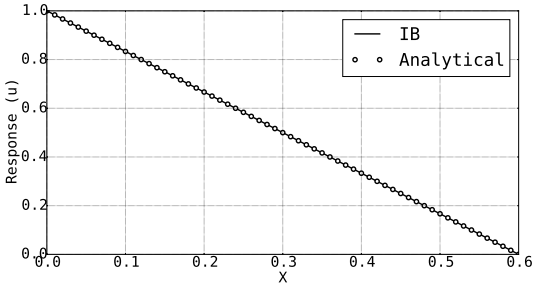
$$(x-1)^2 + (y-2)^2 = 4 \tag{3.17}$$



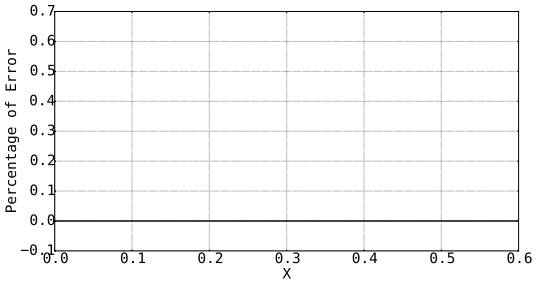
(a) Solution for $X = 0.385$ and moving wall velocity of $1m/s$



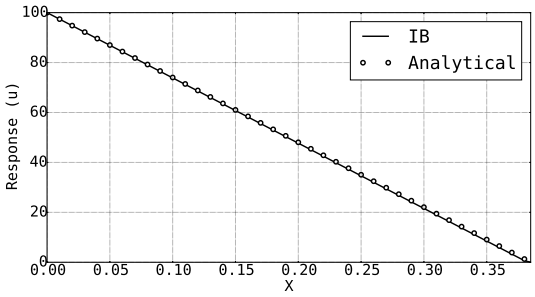
(b) Error plot



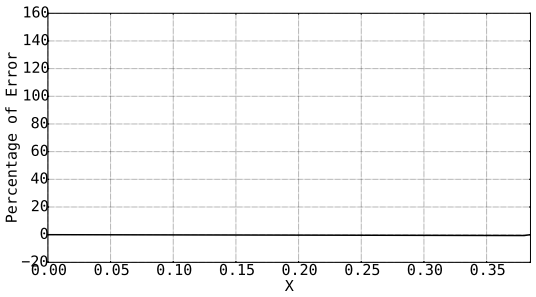
(c) Solution for $X = 0.6$ and moving wall velocity of $1m/s$



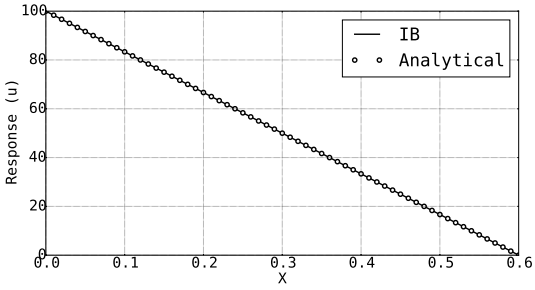
(d) Error plot



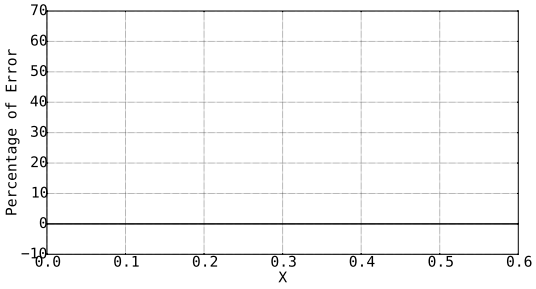
(e) Solution for $X = 0.385$ and moving wall velocity of $100m/s$



(f) Error plot



(g) Solution for $X = 0.6$ and moving wall velocity of $100m/s$



(h) Error plot

Figure 3.4: Comparison between IB and analytical results for different locations of stationary wall.

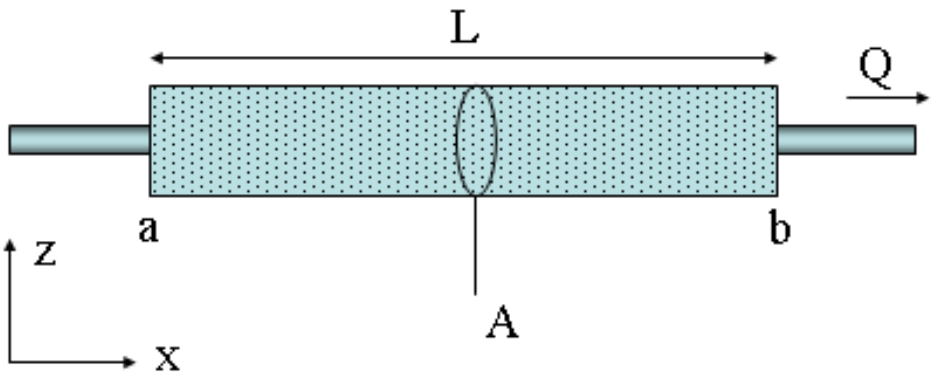


Figure 3.5: Flow through a porous pipe.

Bibliography

[1] 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.

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

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

[52] Cross, D. M., *Local Continuum Sensitivity Method for Shape Design Derivatives Using Spatial Gradient Reconstruction*, Virginia Tech, 2014.

[53] 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.

[54] 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.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

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

[61] 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.

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

[63] 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.