

Fall 2015

Applying vessel inlet/outlet conditions to patient-specific models embedded in Cartesian grids

Aaron Matthew Goddard
University of Iowa

Copyright 2015 Aaron Matthew Goddard

This thesis is available at Iowa Research Online: <https://ir.uiowa.edu/etd/1970>

Recommended Citation

Goddard, Aaron Matthew. "Applying vessel inlet/outlet conditions to patient-specific models embedded in Cartesian grids." MS (Master of Science) thesis, University of Iowa, 2015.
<https://doi.org/10.17077/etd.6yjl67sm>

Follow this and additional works at: <https://ir.uiowa.edu/etd>

Part of the [Biomedical Engineering and Bioengineering Commons](#)

APPLYING VESSEL INLET/OUTLET CONDITIONS TO PATIENT-SPECIFIC
MODELS EMBEDDED IN CARTESIAN GRIDS

by

Aaron Matthew Goddard

A thesis submitted in partial fulfillment
of the requirements for the Master of Science
degree in Biomedical Engineering in the
Graduate College of
The University of Iowa

December 2015

Thesis Supervisor: Associate Professor Sarah C. Vigmstad

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

Aaron Matthew Goddard

has been approved by the Examining Committee for
the thesis requirement for the Master of Science degree
in Biomedical Engineering at the December 2015 graduation.

Thesis Committee:

Sarah C. Vigmostad, Thesis Supervisor

H.S. Udaykumar

Madhavan Raghavan

Edward Sander

Jia Lu

To my wife and family for all their love and support.

ACKNOWLEDGEMENTS

I would like to thank my advisors, Associate Professor Sarah Vigmostad, and Professor H.S Udaykumar for their guidance and opportunity to create this body of work.

I would like to thank my wife, Amanda, and my family for their love and continued support during my education. I would also like to thank lab mates Nirmal, Oishik, Liza, Moustafa, Ehsan, Mike, Sidhartha, Pratik, Kayley, and Jared for creating an enjoyable work environment.

ABSTRACT

Cardiovascular modeling has the capability to provide valuable information allowing clinicians to better classify patients and aid in surgical planning. Modeling is advantageous for being non-invasive, and also allows for quantification of values not easily obtained from physical measurements. Hemodynamics are heavily dependent on vessel geometry, which varies greatly from patient to patient. For this reason, clinically relevant approaches must perform these simulations on patient-specific geometry. Geometry is acquired from various imaging modalities, including magnetic resonance imaging, computed tomography, and ultrasound. The typical approach for generating a computational model requires construction of a triangulated surface mesh for use with finite volume or finite element solvers. Surface mesh construction can result in a loss of anatomical features and often requires a skilled user to execute manual steps in 3rd party software. An alternative to this method is to use a Cartesian grid solver to conduct the fluid simulation. Cartesian grid solvers do not require a surface mesh. They can use the implicit geometry representation created during the image segmentation process, but they are constrained to a cuboidal domain. Since patient-specific geometry usually deviate from the orthogonal directions of a cuboidal domain, flow extensions are often implemented. Flow extensions are created via a skilled user and 3rd party software, rendering the Cartesian grid solver approach no more clinically useful than the triangulated surface mesh approach. This work presents an alternative to flow extensions by developing a method of applying vessel inlet and outlet boundary conditions to regions inside the Cartesian domain.

PUBLIC ABSTRACT

Cardiovascular modeling has the capability to provide valuable information allowing clinicians to better classify patients and aid in surgical planning. Modeling is advantageous for being non-invasive, and also allows for quantification of values not easily obtained from physical measurements. Fluid flows are heavily dependent on vessel geometry, which varies greatly from patient to patient. For this reason, clinically relevant approaches must perform these simulations on individual patient geometry. Geometry is acquired from various imaging methods including MRI, CT, and ultrasound. The typical approach for generating a computational model requires construction of a triangulated surface mesh for use with finite volume or finite element solvers. Surface mesh construction can result in a loss of anatomical features and often requires a skilled user to execute manual steps in 3rd party software. An alternative to this method is to use a Cartesian grid solver to conduct the fluid simulation. Cartesian grid solvers do not require a surface mesh. They can use the geometry representation created during the image segmentation process, but they are constrained to a box shaped domain. Since individual patient geometry usually deviate from the perpendicular directions of a box-shaped domain, flow extensions are often implemented. Flow extensions are created via a skilled user and 3rd party software, rendering the Cartesian grid solver approach no more clinically useful than the triangulated surface mesh approach. This work presents an alternative to flow extensions by developing a method of applying vessel inlet and outlet boundary conditions to regions inside the Cartesian domain.

TABLE OF CONTENTS

LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER 1 INTRODUCTION	1
1.1 Applications of computational modeling in the cardiovascular system.....	1
1.2 Challenges of patient-specific modeling in the clinical environment	3
1.3 Challenges with the conventional approach to Cartesian domain boundaries	5
1.4 Proposed alternative – Interior boundary conditions	6
CHAPTER 2 METHODS	9
2.1 Specific features of the selected computational framework.....	9
2.2 Comparison of Cartesian grid construction.....	10
2.3 Defining interior boundary cap parameters.....	11
2.3.1 Locating the boundary cap endpoints.....	12
2.4 Cell reclassification	13
2.4.1 Ghost Fluid cell classification	14
2.5 Flow solver modifications	14
2.5.1 Time splitting method.....	15
2.5.2 Modification to the gradient scalar calculation	17
2.5.3 Modification to the convection calculation	19
2.5.4 Modification for extrapolation to boundary	21
2.5.5 Modifications for the Dirichlet boundary condition.....	21
2.5.6 Global mass conservation.....	22
2.5.7 Modifications to the outlet boundary condition	23
2.5.8 Modification to the Helmholtz equation.....	24
2.5.9 Modification to the divergence calculation	27
2.5.10 Modifications to the Pressure Poisson equation	29
CHAPTER 3 VALIDATION	43
3.1 Validation of Poiseuille flow.....	43
3.2 Validation of 2D aortic arch with commercial software	44
CHAPTER 4 EMPLOYING INTERIOR BOUNDARY CAPS TO MODEL IMAGE- BASED GEOMETRY	49
4.1 Image-to-flow on patient-specific data	49

4.2 Other image-to-flow uses	49
4.2.1 Image-to-flow on computer drawn image	49
4.2.2 Image-to-flow on hand drawn image.....	50
CHAPTER 5 CONCLUSIONS	53
5.1 Summary	53
5.2 Limitations and future work.....	53
REFERENCES	55

LIST OF TABLES

Table 1: Comparison of boundary cap max to the analytical value.....	47
--	----

LIST OF FIGURES

Figure 1: Example of a typical physiologic geometry which fails to conform to a Cartesian domain. 7

Figure 2: Flow extensions for conventional domain boundaries. (a) 2D representation of idealized aorta geometry (b) Idealized aorta with flow extensions that would be traditionally required to perform flow computation on Cartesian grid solver. 7

Figure 3: Illustration of cells near the desired interior boundary cap being inserted into the conventional domain arrays for purposes of applying vessel inlet/outlet boundary conditions..... 8

Figure 4: Comparison of computational domain construction between (a-d) conventional domain boundary conditions and (e-h) interior boundary caps. (a,e) set up initial grid and insert geometry, (b,f) prune mesh and apply boundary caps, (c,g) refine mesh and classify cells as fluid/solid, (d,h) perform flow simulation. 34

Figure 5: Some of the new geometric configurations interior boundary caps allow. The fluid domain is marked in dark gray. (a) Entire geometry inside the domain, (b) All inlets/outlets crossing domain boundaries, (c) Combination of each, (d) Interior boundary caps used with conventional domain boundary conditions. 35

Figure 6: Comparison of how boundary conditions cells are selected (a-c) conventional domain boundaries (d-f) interior boundary conditions, (a,d) initial grid, (b,e) insert geometry and reclassify outside cells as solid, (c) level set values are used to identify domain boundary cells, (f) level set values and boundary cap plane are used to identify boundary cap cells and cells outside the plane are changed to solid. 36

Figure 7: Differences between conventional domain boundary cells (a-c) and interior boundary cap cells (d-f). (a,d) boundary cell spacing, (b,e) flux contribution to neighbors, (c,f) boundary condition normal..... 37

Figure 8: Linear extrapolation to boundary. Interpolation between the encircled cells is used to determine values of the probe points (solid black circles). 38

Figure 9: Dirichlet boundary condition projection for interior boundary conditions 38

Figure 10: Calculation of mass flux using stair step method..... 39

Figure 11: Helmholtz equation stencil of coefficients for interior cells. 39

Figure 12: Helmholtz equation stencil of coefficients for boundary adjacent cells (a) conventional domain boundaries (b) interior boundary conditions.....	40
Figure 13: Pressure Poisson equation stencil of coefficients for interior cells.	40
Figure 14: Pressure Poisson equation stencil of coefficients for west boundary adjacent cells (a) conventional domain boundaries, (b) typical interior boundary conditions, (c) stair step interior boundary conditions.....	41
Figure 15: Neumann conditions for construction of the Poisson operator matrix at boundary adjacent cells noted by a plus, (a) typical boundary adjacent stencil including a single boundary neighbor, (b) indication of cells needed to apply substitution for boundary cell, (c) boundary adjacent stencil at stair step showing two boundary neighbors, (d) boundary adjacent stencil for stair step cell for substitution of boundary cells.	42
Figure 16: Velocity magnitude comparison ($Re = 100$) using pELAFINT3D with interior boundary caps (top) to pELAFINT3D using conventional domain boundaries (bottom).	45
Figure 17: Poiseuille flow validation at $Re = 100$. (top) longitudinal centerline velocity, (bottom) fully developed radial velocity profile at 8 diameters from the inlet.	46
Figure 18: Velocity magnitude comparison ($Re = 100$) between pELAFINT3D using interior boundary caps (left) and ANSYS® Fluent (right).....	47
Figure 19: Comparison of velocity profiles between pELAFINT3D with interior boundary caps and commercial software Fluent at selected locations.	48
Figure 20: Image to flow simulation on patient specific data. (left) patient image data of descending aorta, (center) level set field generated from image segmentation with interior boundary cap locations marked, (right) velocity magnitude contours at Re of 115 with parabolic inlet velocity profile.....	51
Figure 21: Image-to-flow on computer drawn image. (left) 128 x 128 pixel image, (right) Velocity magnitude contours at $Re = 100$	51
Figure 22: Image-to-flow via scanner. (left) hand drawn image, (right) Velocity magnitude contours at $Re = 100$	52
Figure 23: Image-to-flow via cellphone camera. (left) hand drawn image, (right) Velocity magnitude contours at $Re = 100$	52

CHAPTER 1 INTRODUCTION

1.1 Applications of computational modeling in the cardiovascular system

Computational modeling in the cardiovascular system is an active area of research. Modeling capabilities in this field continue to be developed and enhanced, with efforts focusing on enabling advancements in the ability to understand disease development, progression, and treatment pathways. Most of the applications can be grouped into five general categories.

First, consider computational modeling for the understanding of disease development and progression. An example of this is Groen et al. in [1] where computational fluid dynamics is used to correlate wall shear stresses with plaque composition and rupture. Magnetic Resonance Imaging (MRI) data was used to construct a volumetric mesh for simulation purposes. Upon completing these simulations they had evidence to conclude that high wall shear stresses influence plaque vulnerability and may be a reliable predictor of rupture.

Computational modeling can be used to identify and evaluate alternative treatments. Esmaily-Moghadam et al. present an exciting example of this in [2] where they evaluate alternative surgical treatments for hypoplastic left heart syndrome. The traditional treatment plan consists of three surgical procedures to correct this congenital defect. Computational modeling was used to investigate alternatives to the traditional approach. The simulations were performed on idealized geometric models. The models were volumetrically meshed to perform a finite element simulation with lumped parameter boundary conditions. The insights gained from these idealized models have

resulted in the group identifying a promising two-step procedure to replace the traditional approach.

Research and development of new medical devices is another area where computational tools have the potential for large impact. For example, Borazjani et al. in [3] use a curvilinear immersed boundary method to simulate the fluid structure interaction (FSI) of a bi-leaflet mechanical heart valve operating through the cardiac cycle. The motivation for doing so is to better design mechanical heart valves to minimize thrombus formation, which is thought to be caused by the high fluid shear stresses near the valve hinge regions. Putting a tool such as this in the hands of research engineers at medical device companies, would likely produce better quality products resulting in improved patient outcomes.

Computation modeling can be used to make non-invasive measurements and aid in patient classification. For example, Taylor et al. in [4] have developed the capability to evaluate Fractional Flow Reserve (FFR) in patients with coronary occlusion. FFR has been shown to be a useful index for determining the severity of coronary stenosis [5]. Starting from patient-specific Computed Tomography (CT) data, Taylor et al. are able to extract patient geometry, create a volumetric mesh, apply lumped-parameter boundary conditions, and solve for FFR using the Finite Element Method (FEM). The measurement of FFR is traditionally performed via catheter, thus computational modeling enables a technique to obtain these measurements in a less-invasive manner.

Finally, computational modeling can also be used for patient-specific treatment planning. Xiong et al. in [6] have developed a method of assessing and optimizing patient-specific outcomes by simulating medical interventions of stent and graft

deployment. Patient image data is reconstructed as a triangular surface mesh. The geometric mesh is cut and spliced to replace the diseased portion of the vessel with the treatment geometry, allowing for the simulation and prediction of the proposed treatment. When developed into a clinical tool, this will allow for clinicians to tailor the treatment to the individual patient.

The latter two applications described above have focused on the development of patient-specific modeling within the clinical environment. It should be evident that a simulation tool developed for the clinical environment requires specific features. It should be a single software platform and require minimal data transfer/translation. It should also be designed for ease of use with minimal training or user specific knowledge required to conduct the simulation. Methods intended to streamline this process or decrease user burden are integral to bringing these applications to fruition.

1.2 Challenges of patient-specific modeling in the clinical environment

As described above, numerical simulations have been shown to be a useful tool in patient classification and surgical planning [4], [6], [7]. Variation in patient parameters and geometry can have significant effects on the cardiovascular hemodynamics, thus individual patient simulation is the ideal approach [8]. Patient geometry may be acquired through a variety of non or minimally-invasive imaging techniques including, magnetic resonance imaging (MRI), computed tomography (CT), and ultrasound. The process of performing simulations from these images follows a typical set of steps [6], [7], [9]. The image is first filtered and segmented, for which the levelset approach is commonly used [10], [11]. Often a marching cubes method is used to convert this implicit surface representation into an explicit triangular surface mesh [12]. Depending on the specific

patient geometry, several additional mesh processing steps are performed including mesh smoothing, trimming, elongation, refining and remeshing. This surface mesh is then used to construct a volumetric mesh. Finally, a finite volume or finite element method is used to perform the computational fluid dynamic (CFD) simulation.

While software packages such as Mimics® and VMTK have become available to aid in the construction of the surface/volumetric meshes, the process still requires significant intervention from a skilled user. Often the flow computations are performed in a software different from which the mesh is generated. These two factors render this traditional process as too burdensome for the clinical environment.

Cartesian grid CFD solvers offer an alternative solution by eliminating the need for construction of surface or volumetric meshes. This process begins in a similar fashion to the traditional approach previously described. As before, the image is first filtered and segmented using the level set method. However, rather than using the level set field to generate a surface mesh, level set values can directly map onto the Cartesian grid domain using an immersed boundary type method to apply boundary conditions at the vessel walls. Image-to-flow computation without a volumetric mesh is an attractive solution which has spurred many groups to pursue this capability [13]–[16].

While the advantages of image-to-flow computation on a Cartesian grid are substantial, there exists a burdensome restriction that has yet to be sufficiently resolved. Cartesian grid solvers operate by embedding an implicit representation of the geometry inside a rectangular or cuboidal domain. All inlet and outlet boundary conditions are thus applied to the domain boundary (sides of the rectangular or cuboidal domain). This means all vessels must extend to a side of the domain, crossing it in a perpendicular

orientation. Unfortunately, physiologic geometry is tortuous and does not typically conform to these burdensome restrictions. Despite efforts to selectively rotate and crop geometry, much of the patient data will not conform to a rectangular or cuboidal domain. The top face of Figure 1 is a clear example of patient geometry that doesn't meet the necessary requirements. The vessel crossing the top face does so at an angle other than 90 degrees. A boundary condition applied to this vessel has the potential to create undesired entrance effects.

1.3 Challenges with the conventional approach to Cartesian domain boundaries

In Cartesian domains, flow extensions can be used to incorporate a “nonconforming” vessel within a rectangular/cuboidal domain. For simplicity, consider the idealized 2D aorta representation shown in Figure 2a. An example of the flow extensions needed to compute this geometry using a Cartesian grid solver is shown in Figure 2b. Flow extensions have multiple major drawbacks. First, non-physiologic vessel curvature is often introduced to align the ends of the vessel perpendicular to the domain. This added curvature has the potential to introduce unwanted secondary flows thus decreasing the accuracy of the simulation. Second, the added cells increase the domain size, and thus the computation time. Third, flow extensions limit the locations where boundary conditions can be applied. This is demonstrated by considering the original geometry shown in Figure 2. To perform an accurate simulation of flow through the aorta, one would apply a known velocity profile, as described in [17], to the inlet at the aortic root as denoted by AR in Figure 2a. The measured velocity profiles should be applied as the inlet boundary condition at location 1 of Figure 2b, but due to the addition of the flow extension, the velocity profile can only be applied to location 2. The fourth

and most significant drawback of flow extensions, is the manual process required to construct them. A skilled user must apply his or her best judgement in creating appropriate flow extensions using a third party software. Using multiple software applications with manual input of a skilled user is in conflict with the goal of developing a tool that can be used in the clinical environment. Flow extensions are sometimes used for reasons other than conformance to the Cartesian domain. They can be used to generate a fully developed velocity profile from a uniform velocity inlet. Directly applying the velocity profile is a better alternative due to the computational savings.

1.4 Proposed alternative – Interior boundary conditions

This work presents an alternative to flow extensions by applying boundary conditions to the interior of the rectangular domain rather than the conventional approach of applying them to the sides of the rectangular domain. The general approach combines user defined interior boundary cap locations with the implicit geometric representation in order to identify interior cells that will be reclassified as boundary cells. These interior cells are then inserted into the conventional domain boundaries for computational purposes. This concept is illustrated in Figure 3. Multiple modifications must be made to the flow solver to accommodate the difference between the interior boundary cells and the domain boundary cells to which boundary conditions are traditionally applied. Additional flow solver modifications are necessary to allow the interior boundary conditions to be supplied at an orientation other than perpendicular to the domain boundary. The method developed and implemented is described in Chapter 2, with validation studies and applications of this technique demonstrated in subsequent chapters.

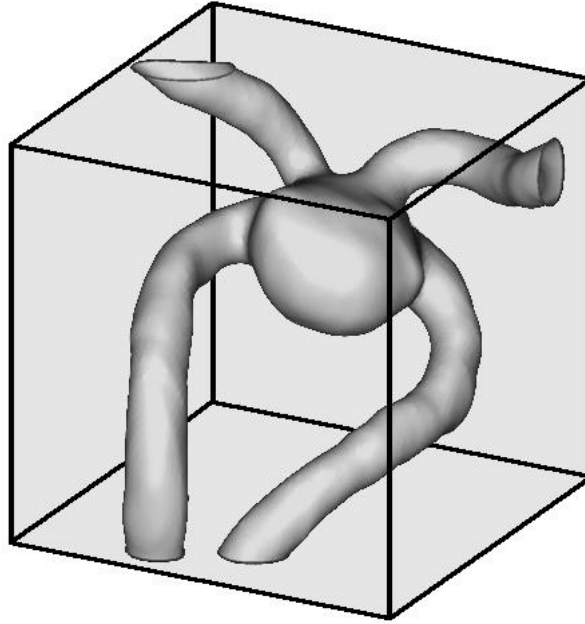


Figure 1: Example of a typical physiologic geometry which fails to conform to a Cartesian domain.

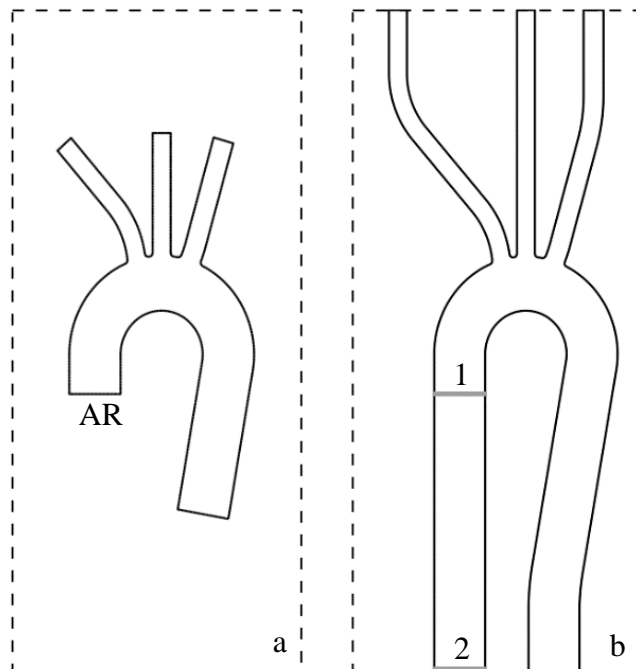


Figure 2: Flow extensions for conventional domain boundaries. (a) 2D representation of idealized aorta geometry (b) Idealized aorta with flow extensions that would be traditionally required to perform flow computation on Cartesian grid solver.

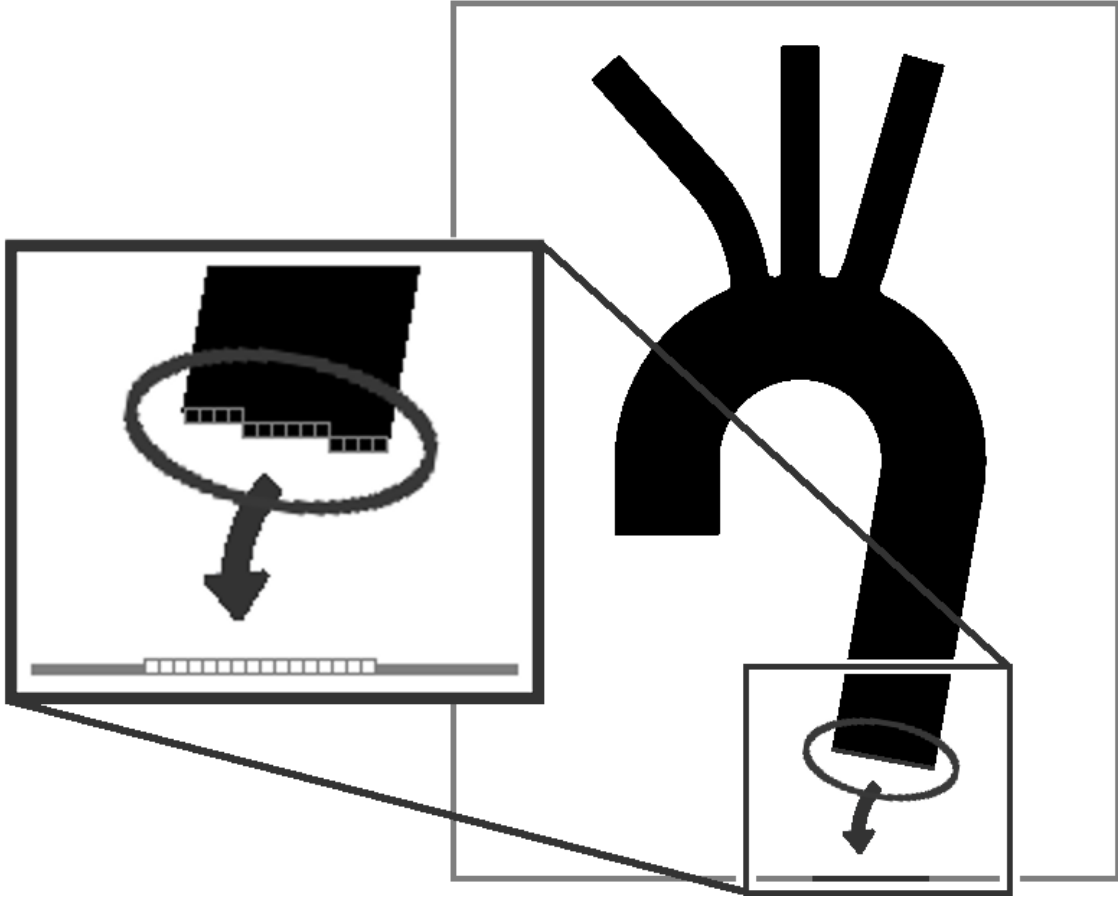


Figure 3: Illustration of cells near the desired interior boundary cap being inserted into the conventional domain arrays for purposes of applying vessel inlet/outlet boundary conditions.

CHAPTER 2 METHODS

To implement the method of interior boundary caps, one must consider all aspects of the computational framework which may be affected. First the specific features and capabilities of the computational framework must be considered. For example, flow solvers which have the ability to automatically refine the computational mesh must be handled differently from those which do not. Next, the parameters needed to fully define the interior boundary cap location, orientation, mapping, etc. must be identified. Then, construction of the Cartesian domain must be examined for required changes. Cell classifications per the features of the flow solver must be also be considered. Finally, the discretization schemes used to perform the CFD computation must be examined and modified accordingly.

2.1 Specific features of the selected computational framework

While this approach can be applied to many Cartesian based CFD solvers, it has been developed with respect to pELAFINT3D. pELAFINT3D is a software package specializing in simulation of Fluid-Structure Interaction (FSI) problems of incompressible fluid flow. The boundary conditions from the immersed geometry are communicated to the flow equations via a sharp interface approach which directly incorporates the boundary conditions into the discretization operators near the interface [18]. Multiple sharp interface methods have been developed. This framework employs the Ghost Fluid Method (GFM) [19]–[21]. It utilizes a four step pressure correction scheme to solve the incompressible Navier-Stokes equations [22]–[24]. pELAFINT3D is massively parallel with the partitioned domain being load balanced at regular intervals [25]. Mesh pruning has been implemented to remove cells far from the region of interest

for computational efficiency. Automatic Local Mesh Refinement (LMR) is used to increase the simulation accuracy while minimizing the computational burden [25], [26]. The immersed geometry is represented via level set function. This package also includes the capability to filter and segment the patient images allowing for a complete image-to-flow computation [27].

2.2 Comparison of Cartesian grid construction

To clearly describe the method of applying internal boundary caps, it is helpful to identify the steps employed to create the Cartesian grid, and embed a geometry of interest within the domain, represented using level sets. First consider the case for conventional boundary conditions shown in Figure 4a-d. Initially a rectangular or cuboidal domain is created to a particular size and cell spacing per user specification. The geometry is then immersed in the Cartesian grid (a). For the image-to-flow application, the geometry is inserted by mapping the level set field created from image segmentation. For the case of conventional domain boundaries, the geometry is restricted to having all vessels extend to and align perpendicular to the sides of the domain. Next the initial base mesh is pruned to remove cells far outside the geometry resulting in significant computational savings (b). LMR then automatically refines the mesh via octree method [26]. The cells near the vessel wall are the smallest refinement level allowed (c). Level set values are then used to determine which cells are inside the fluid domain and then flagged accordingly. Finally, the flow solver is able to perform the flow computations (d).

This process is slightly altered when implementing the internal boundary end caps (Figure 4e-h). The initial creation of the rectangular or cuboidal domain boundary remains the same. Now, however, the geometry can be placed into the domain with

fewer restrictions (e). In the same manner as before, the cells far outside the geometry are then pruned from the computational domain. An additional step defines the interior boundary caps from user specified values (f). LMR proceeds in a similar manner by refining near the vessel walls, but also refines to the lowest allowed level near the boundary cap (g). Fluid cells are now determined by two criteria. These cells must have a positive level set and must lie in the region between the boundary caps. Thus, not all cells with a positive level set value will be considered part of the fluid domain. The flow solver is then able to conduct the flow simulation (h).

Since conventional domain boundaries are applied to the sides of the cuboidal domain, they have the advantage of being oriented in the direction of the Cartesian rows and columns. It would be simple to implement interior boundary caps in those same orientations, but this would not provide the flexibility needed to accommodate the variability of physiologic geometry. For this reason, the current framework has been developed to support all of the following configurations shown in Figure 5. The inlet/outlet vessels can (a) end short of the domain boundary, (b) cross the domain boundary, or (c) any combination of the two. Also, vessel orientation is no longer a restriction. It should also be evident that the geometry can now be oriented in any manner. Figure 5d shows how conventional domain boundary conditions may also be combined with interior boundary caps.

2.3 Defining interior boundary cap parameters

To perform a simulation utilizing interior boundary caps, additional parameters are needed. The current framework requires three additional inputs for each boundary condition. First is the Cartesian coordinates of the point where the vessel is to be cut.

Second is a unit vector defining the orientation of the boundary cap cut plane. The current convention is for the vector pointing toward the fluid domain. Third is the side of the Cartesian domain the boundary cap is to be mapped onto. For the current framework, all of these parameters are defined by the user.

2.3.1 Locating the boundary cap endpoints

After construction of the initial base mesh, the first step of implementation is to locate the boundary cap endpoints. These are the locations where the boundary cap plane crosses the zero level set contour. All points in the zero level set array are tested for proximity to the plane defined by the cut point and the normal vector (boundary cap plane). This framework utilizes a narrow level set tube approach with the cells adjacent to the interface populated into the zero level set array [10]. Tortuous vessels may cross the boundary cap plane more than once resulting in extra zero level set cells being identified. For the 2D case, only the two closest zero level set points are used to construct the boundary cap end points. Neighbors of each are used with a least squares method to construct a line representing the interface at that location. The intersection of the line and the boundary cap plane is taken to be the boundary cap end point. This newly defined interior boundary cap region (between the two end points) is used as criteria for the adaptive mesh refinement. Cells within a prescribed distance from the boundary cap region are refined to the smallest cell size allowed. We have found a distance of twice the base cell size to be sufficient refinement criteria.

2.4 Cell reclassification

When creating the initial domain, the conventional domain boundary cells are already known from construction of the Cartesian base mesh. Upon inserting the vessel geometry, these cells are tested for a positive level set value where the vessel crosses the cuboidal domain boundary. This process of conventional domain boundary cell selection and fluid cell reclassification is illustrated in Figure 6(a-c), with Figure 6(d-f) showing the steps for interior boundary caps. First the initial grid is constructed Figure 6(a,d). The geometry is inserted, such that cells outside the geometry are classified as solid, denoted as hollow circles Figure 6(b,e). Conventional domain boundaries then use only the level set value to determine which cells are to be used for applying boundary conditions, marked by X's Figure 6(c). Both the level set and interior boundary cap plane are used to identify boundary cap cells. A search algorithm is implemented to identify cells just inside of the boundary cap plane and inside the vessel. This will result in a stair step arrangement of interior boundary cap cells to which vessel inlet or outlet boundary conditions will be applied. Cells in the domain boundary arrays are removed and the boundary cap cells are populated instead. Cells outside of the boundary cap plane, regardless of their level set value, are reclassified as solid Figure 6(f).

Tortuous geometry may result in a boundary cap plane which crosses another region of the vessel. The inlet at the aortic root (AR) from Figure 2(a) is an example of this. The boundary cap plane defining this inlet also crosses the descending aorta. Thus, an additional criteria must be used to determine which cells should no longer be part of the fluid domain. This specific approach limits cell reclassification to a distance beyond

the endpoints of the boundary cap line. Twice the level set tube thickness has been found to be sufficient criteria.

2.4.1 Ghost Fluid cell classification

Since the current framework utilizes GFM for the immersed boundary conditions, GFM classification of cells must also be considered for the use of interior boundary caps [28]. Computational cells are given one of five classifications: (1) Fluid cells are located in the fluid domain far from an interface; (2) Hybrid cells are in the fluid domain, but are in close proximity to the solid interface. These cells are given special treatment to reduce pressure oscillations when the boundary is moving. Only stationary geometry is currently being considered; therefore, the hybrid cells will be treated as if they are fluid cells; (3) Solid cells are far outside of the fluid domain and are thus not included in the GFM computations; (4) Ghost cells are also outside the fluid domain, but have a neighbor inside the fluid domain. It is these cells which will have a ghost value computed for use in the finite difference schemes; (5) Finally, boundary cells are ones to which domain boundary conditions are applied. All cells selected for the interior boundary cap will be reclassified to GFM boundary type. Also, all fluid or hybrid cells which now lie on the outside of the interior boundary cap plane are reclassified to GFM solid type.

2.5 Flow solver modifications

Conventionally, domain boundary conditions are applied to a layer of zero thickness cells constructed on the surface of the cuboidal domain (domain boundary cells). The method of interior boundary caps instead applies the constraints to cells inside the domain, which can be oriented in any direction, and thus are likely to be

arranged in a stair-step manner. Figure 7 shows the three main differences between these cells. The flow solver must be modified to accommodate these differences. First is cell spacing (Figure 7a,d). Since the domain boundary cells have zero thickness, the cell centers lie directly on the face of the first interior cell resulting in the distance between the cell centers being only half the cell width. Everywhere inside the domain, the distance between cell centers is the full width of the cell, which is the case for the interior boundary cap cells. Finite difference schemes must be modified for every step of the flow solver. Second is the boundary cell contribution (Figure 7b,e). Conventional domain boundary cells lie on the surface of the cuboidal domain and thus only contribute a flux to one interior neighbor. This is also true for many of the boundary cap cells. When the boundary cap is not parallel to the cuboidal domain, boundary cap cells will be selected in a stair step manner. At each stair step the last boundary cap cell will have two interior neighbors, thus contributing a flux to each. Finally the boundary orientation must be considered (Figure 7c,f). Conventional domain boundaries are always in the direction of the Cartesian grid. All Neumann boundary conditions or extrapolations simply use the cells in that particular row or column. Since interior boundary caps are not necessarily oriented in a Cartesian direction, the cells needed to calculate Neumann boundary conditions or extrapolations may include cells from other rows and columns.

2.5.1 Time splitting method

To outline the changes to the flow solver due to these differences, the specifics of the four step pressure correction algorithm must be described. Sub-iterations are used to march the solution forward in time. The starting point is the incompressible Navier-Stokes equations for a Newtonian fluid. The non-dimensional form of these equations is

shown below with \mathbf{u} and p denoting the non-dimensional velocity and pressure fields. U and L are the characteristic length and velocity scales of the problem with ν as the constant kinematic viscosity.

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla p + \frac{\nu}{UL} \nabla^2 \mathbf{u} \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (2.2)$$

A four step pressure correction scheme is used to solve these equations. First, a provisional velocity denoted by \mathbf{u}^* is calculated from the momentum equation. The time derivative is computed using a second order backwards differencing scheme. The convection and pressure gradient terms are explicitly computed from the most recent values available, either the previous sub-iteration or the previous time step. This results in the following representation of the momentum equation where the n superscript denotes the time step, k denotes the sub-iteration and α is a scalar value based on the time step sizes.

$$(\alpha_1 \mathbf{u}^* + \alpha_2 \mathbf{u}^n + \alpha_3 \mathbf{u}^{n-1}) + ((\mathbf{u} \cdot \nabla) \mathbf{u})_k^{n+1} = -\nabla p_k^{n+1} + \nu \nabla^2 \mathbf{u}^* \quad (2.3)$$

All of the explicit terms can be moved to the right hand side resulting in the following Helmholtz equation.

$$\alpha_1 \mathbf{u}^* - \nu \nabla^2 \mathbf{u}^* = -\nabla p_k^{n+1} - \alpha_2 \mathbf{u}^n - \alpha_3 \mathbf{u}^{n-1} - ((\mathbf{u} \cdot \nabla) \mathbf{u})_k^{n+1} \quad (2.4)$$

Before solving the Helmholtz equation the explicit convection and pressure gradient terms must be evaluated. Both of these computations must be modified to accommodate the implementation of interior boundary conditions.

2.5.2 Modification to the gradient scalar calculation

First consider the gradient of a scalar calculation. For this instance the scalar field is pressure. The pressure values from the previous sub iteration will be used to perform this calculation. The term is expanded as follows.

$$\nabla\phi = \left[\frac{d\phi}{dx} \quad \frac{d\phi}{dy} \quad \frac{d\phi}{dz} \right] \quad (2.5)$$

First order central differencing can be used to approximate these derivative values. This finite difference scheme is 2nd order accurate.

$$\frac{d\phi}{dx} = \frac{\phi_{i+1} - \phi_{i-1}}{2dx} \quad (2.6)$$

For clarity, notations defined below can replace the subscript indices with the cell neighbor direction.

$$\left[\begin{array}{l} \phi_{i+1,j} = \phi_E \\ \phi_{i-1,j} = \phi_W \\ \phi_{i,j+1} = \phi_N \\ \phi_{i,j-1} = \phi_S \\ \phi_{i,j} = \phi_{IC} \end{array} \right] \begin{array}{l} EAST \\ WEST \\ NORTH \\ SOUTH \\ CURRENT \end{array} \quad (2.7)$$

The finite difference equation is rewritten below in terms of cell neighbors.

$$\frac{d\phi}{dx} = \frac{\phi_E - \phi_W}{2dx} \quad (2.8)$$

This can also be expressed in terms of the values at the cell faces.

$$\frac{d\phi}{dx} = \frac{\frac{\phi_E + \phi_{IC}}{2} - \frac{\phi_{IC} + \phi_W}{2}}{dx} \quad (2.9)$$

If either of these faces lie on the Cartesian domain boundary, the cell spacing is not a full cell (dx) away. The finite difference scheme must be updated accordingly. This condition is illustrated in Figure 7a. Domain boundary cells are positioned on the cell face, so those values should directly be substituted into the finite difference scheme as shown below for a west domain boundary.

$$\left. \frac{d\phi}{dx} \right|_{Wbdry} = \frac{\frac{\phi_E + \phi_{IC}}{2} - \phi_W}{dx} \quad (2.10)$$

For convenience Equation 2.10 can be split into two arguments as shown below. The first will be executed on all domain interior cells, and the second term will only be executed on boundary adjacent cells to account for the half cell spacing. The equation below shows this formulation for a west boundary with only the direction indices changing for the other boundary directions.

$$\left. \frac{d\phi}{dx} \right|_{Wbdry} = \frac{\frac{\phi_E + \phi_{IC}}{2} - \frac{\phi_{IC} + \phi_W}{2}}{dx} + \frac{\frac{\phi_{IC} - \phi_W}{2}}{dx} \quad (2.11)$$

For computational efficiency the first term is split as shown below for the x direction.

$$\left. \frac{d\phi}{dx} \right|_{Wbdry} = \frac{\frac{\phi_E + \phi_{IC}}{2}}{dx} - \frac{\frac{\phi_{IC} + \phi_W}{2}}{dx} + \frac{\frac{\phi_{IC} - \phi_W}{2}}{dx} \quad (2.12)$$

In the x direction the current cell will share the first term with the west neighbor and share the second term with the east neighbor. Rather than duplicating every one of these computations, the current cell only calculates the second term. The value is

summed as a negative contribution to the current cell and summed as a positive term to the east neighbor. Each cell will thus receive the first term from the calculation of the west neighbor. After this calculation has been completed on all interior cells, the boundary adjacent cells of the west and south boundaries will be missing the neighbor contribution of the first term in Equation 2.12. The conventional domain approach loops through the domain boundary cells on the west and south sides. This allows the those domain boundary cells to contribute the missing first term to a single interior cell as shown in Figure 7b. Since interior boundary cap cells are often arranged in a stair step manner (Figure 7e), the boundary cap cells at the stair step may need to contribute this term to east and north neighbors.

Finally, the conventional approach will loop through all of the domain boundary cells to contribute the third term of Equation 2.12 to the interior neighbor. Since interior boundary cap cells are a full cell width as shown in Figure 7d, the third term is ignored.

2.5.3 Modification to the convection calculation

Next consider the convection term. This term is computed using the velocity values from the previous sub-iteration or time step and can be expanded as shown below.

$$(\mathbf{u} \cdot \nabla) \mathbf{u} = \left(u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} \right) \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \\ u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \end{bmatrix} \quad (2.13)$$

For computational efficiency the calculation is broken in half and evaluate at each face of the cell.

$$\frac{du}{dx} = \frac{u_E - u_W}{2dx} = \frac{u_E + u_{IC}}{2dx} - \frac{u_{IC} + u_W}{2dx} \quad (2.14)$$

$$\frac{du}{dy} = \frac{u_N - u_S}{2dy} = \frac{u_N + u_{IC}}{2dy} - \frac{u_{IC} + u_S}{2dy} \quad (2.15)$$

$$\frac{dv}{dx} = \frac{v_E - v_W}{2dx} = \frac{v_E + v_{IC}}{2dx} - \frac{v_{IC} + v_W}{2dx} \quad (2.16)$$

$$\frac{dv}{dy} = \frac{v_N - v_S}{2dy} = \frac{v_N + v_{IC}}{2dy} - \frac{v_{IC} + v_S}{2dy} \quad (2.17)$$

Each velocity term at the relevant faces is derived below.

$$u_{(EF)} = \frac{u_E + u_{IC}}{2} \quad (2.18)$$

$$u_{(WF)} = \frac{u_W + u_{IC}}{2} \quad (2.19)$$

$$v_{(NF)} = \frac{v_N + v_{IC}}{2} \quad (2.20)$$

$$v_{(SF)} = \frac{v_S + v_{IC}}{2} \quad (2.21)$$

Each of these terms can be populated in the matrix of above.

$$\left[\begin{array}{l} \left(\frac{u_E + u_{IC}}{2} \frac{u_E + u_{IC}}{2dx} - \frac{u_W + u_{IC}}{2} \frac{u_{IC} + u_W}{2dx} \right) + \left(\frac{v_N + v_{IC}}{2} \frac{u_N + u_{IC}}{2dy} - \frac{v_S + v_{IC}}{2} \frac{u_{IC} + u_S}{2dy} \right) \\ \left(\frac{u_E + u_{IC}}{2} \frac{v_E + v_{IC}}{2dx} - \frac{u_W + u_{IC}}{2} \frac{v_{IC} + v_W}{2dx} \right) + \left(\frac{v_N + v_{IC}}{2} \frac{v_N + v_{IC}}{2dy} - \frac{v_S + v_{IC}}{2} \frac{v_{IC} + v_S}{2dy} \right) \end{array} \right] \quad (2.22)$$

The terms can then be arranged by the direction of contribution with the east and north terms collected to the left and the west and south collected to the right.

$$\left[\begin{array}{l} \left(\frac{u_E + u_{IC}}{2} \frac{u_E + u_{IC}}{2dx} + \frac{v_N + v_{IC}}{2} \frac{u_N + u_{IC}}{2dy} \right) - \left(\frac{u_W + u_{IC}}{2} \frac{u_{IC} + u_W}{2dx} + \frac{v_S + v_{IC}}{2} \frac{u_{IC} + u_S}{2dy} \right) \\ \left(\frac{u_E + u_{IC}}{2} \frac{v_E + v_{IC}}{2dx} + \frac{v_N + v_{IC}}{2} \frac{v_N + v_{IC}}{2dy} \right) - \left(\frac{u_W + u_{IC}}{2} \frac{v_{IC} + v_W}{2dx} + \frac{v_S + v_{IC}}{2} \frac{v_{IC} + v_S}{2dy} \right) \end{array} \right] \quad (2.23)$$

For computational efficiency this equation can be split in the same manner as for the gradient scalar calculation above. The terms on the left are summed as a positive value for the current cell. Those same terms will be contributed to the east and north

neighbors as a negative value accordingly. Therefore, the current cell will receive the terms on the right side of this matrix from the west and south neighbors. As before boundary cells on the west and south domain boundaries must be computed to include the missing term to those boundary adjacent cells. Again when implementing interior boundary caps, the boundary cell at a south-west stair step must contribute flux values to two cells.

2.5.4 Modification for extrapolation to boundary

To complete the calculation of the convection field, values must be extrapolated to the boundary cells. Conventional domain boundaries utilize a 1st order forward difference. Since conventional domain boundaries are orthogonal to the Cartesian grid, as shown in Figure 7c, the finite difference scheme requires only the two adjacent neighbors in the domain normal direction. Interior boundary conditions often have a boundary normal which is non-orthogonal to the domain. This condition is illustrated in Figure 7f. To construct an equivalent linear extrapolation, a total of four neighboring cells are needed. This is shown in Figure 8. The boundary normal vector is used to project the boundary cap cell location to the next two adjacent cell rows. The two probe points are given values by linear interpolation of the two adjacent cells in each row. These two probe values are then used to for linear extrapolation of value to the interior boundary cap cell.

2.5.5 Modifications for the Dirichlet boundary condition

Now that all terms on the right hand side of Equation 2.4 are known, the Helmholtz equation can then be solved. To do so, the boundary conditions must first be applied. Dirichlet boundary conditions are applied directly to the boundary cap cells

even though they do not lie exactly on the boundary cap plane. The velocity vector must be rotated to the orientation of the boundary cap using the equation below where \mathbf{u}_{Domain} is the inlet velocity as would be applied to the side of the Cartesian domain and \mathbf{u}_{Bcap} is the inlet velocity vector rotated to the orientation of the interior boundary cap.

$$\mathbf{u}_{Domain} = \mathbf{Q}\mathbf{u}_{Bcap} \quad (2.24)$$

\mathbf{Q} is an orthogonal rotation tensor that can be constructed from the Cartesian domain basis and the interior boundary cap basis as shown below for the 2 dimensional case. The superscript bar denotes the basis vector of the interior boundary cap.

$$[\mathbf{Q}] = \begin{bmatrix} \bar{e}_1 \cdot e_1 & \bar{e}_1 \cdot e_2 \\ \bar{e}_2 \cdot e_1 & \bar{e}_2 \cdot e_2 \end{bmatrix} \quad (2.25)$$

In the case of a user defined inlet velocity profile the boundary cell locations are projected to the boundary cap line using the following equation. \mathbf{x} is the location vector of the boundary cell with $\mathbf{p1}$ and $\mathbf{p2}$ representing the location vector of the boundary cap end points. This projection is illustrated in Figure 9.

$$Projected\ Distance = \frac{(\mathbf{x}-\mathbf{p1}) \cdot (\mathbf{p2}-\mathbf{p1})}{\|\mathbf{p2}-\mathbf{p1}\|} \quad (2.26)$$

Cubic spline interpolation is used to obtain the velocity value at the projected distance. The interpolated value is transformed to the orientation of the boundary cap in the same way as before by using Equation 2.24.

2.5.6 Global mass conservation

For this framework, conservation of mass is enforced globally. The total mass flux is computed at every inlet and outlet. Since the interior boundary cap case is likely

not pointed in a Cartesian direction, a stair-step approach is used to sum the mass flux at the interior boundary cap cells. Only the component of velocity normal to the stair-step boundary are used for this computation. This mass flux scheme is illustrated below in Figure 10. Any necessary correction is applied to the vessel outlet boundary conditions. Since the mass flux correction is summed in the boundary normal direction (Figure 7f), the plug velocity correction must be decomposed into the x and y velocity components. This is accomplished with the equation below.

$$\mathbf{u}_{Correction} = \mathbf{Q}^T \mathbf{u}_{\Delta Flux} \quad (2.27)$$

\mathbf{Q} is the same rotation tensor as previously defined. $\mathbf{u}_{\Delta Flux}$ is the discrepancy of bulk velocity between the inlets and outlet with $\mathbf{u}_{Correction}$ being the correction velocity applied to all boundary cells of the outlet.

2.5.7 Modifications to the outlet boundary condition

This framework uses an advection equation to apply a mass conserving Dirchlet boundary condition shown below where \bar{U} is the bulk velocity through the outlet and \mathbf{n} is a unit vector normal to interior boundary cap pointing in the direction of flow.

$$\frac{\partial \mathbf{u}}{\partial t} + \bar{U}(\nabla \mathbf{u})\mathbf{n} = 0 \quad (2.28)$$

For this equation to be evaluated in the Cartesian basis, the bulk velocity must be rotated using the same transformation from Equation 2.27. A second order backward difference scheme is used to approximate the time derivative.

2.5.8 Modification to the Helmholtz equation

All terms on the right hand side of the Helmholtz equation (Equation 2.4) are known. The convection term, the pressure gradient, and the velocity values from the two previous time steps can be combined into a single array. This results in the following Helmholtz type equation where \mathbf{b} is the sum of the known values.

$$\alpha_1 \mathbf{u}^* - \nu \nabla^2 \mathbf{u}^* = \mathbf{b} \quad (2.29)$$

Since this equation is a second order PDE, it is beneficial to put it in the form of a system of linear equations as shown below, where \mathbf{A} is the operator matrix, \mathbf{x} is the vector of unknowns, and \mathbf{b} is the solution vector.

$$\mathbf{Ax} = \mathbf{b} \quad (2.30)$$

The Helmholtz equation can be expanded as shown below, expressing the second derivatives of velocity, where u and v are the velocities in the x and y directions respectively. For simplicity, this is demonstrated for the 2-dimensional case.

$$\begin{bmatrix} \alpha u^* - \nu \left(\frac{\partial^2 u^*}{\partial x^2} + \frac{\partial^2 u^*}{\partial y^2} \right) \\ \alpha v^* - \nu \left(\frac{\partial^2 v^*}{\partial x^2} + \frac{\partial^2 v^*}{\partial y^2} \right) \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad (2.31)$$

The second derivative terms can be represented by the second order accurate central differencing scheme shown below.

$$\frac{\partial^2 u^*}{\partial x^2} = \frac{u_{i+1,j}^* - 2u_{i,j}^* + u_{i-1,j}^*}{dx^2} \quad (2.32)$$

$$\frac{\partial^2 u^*}{\partial y^2} = \frac{u_{i,j+1}^* - 2u_{i,j}^* + u_{i,j-1}^*}{dy^2} \quad (2.33)$$

Converting these into the Cartesian neighbor indexing per Equation 2.7 yields the following notation.

$$\frac{\partial^2 u^*}{\partial x^2} = \frac{u_E^* - 2u_{IC}^* + u_W^*}{dx^2} \quad (2.34)$$

$$\frac{\partial^2 u^*}{\partial y^2} = \frac{u_N^* - 2u_{IC}^* + u_S^*}{dy^2} \quad (2.35)$$

Substituting these expressions into Equation 2.31, results in the following system of equations.

$$\begin{bmatrix} \alpha u_{IC}^* - \nu \left(\frac{u_E^* - 2u_{IC}^* + u_W^*}{dx^2} + \frac{u_N^* - 2u_{IC}^* + u_S^*}{dy^2} \right) \\ \alpha u_{IC}^* - \nu \left(\frac{v_E^* - 2v_{IC}^* + v_W^*}{dx^2} + \frac{v_N^* - 2v_{IC}^* + v_S^*}{dy^2} \right) \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad (2.36)$$

Finally, these equations can be simplified to the following form. This result can be represented as a stencil of coefficients shown in Figure 11.

$$\begin{bmatrix} \alpha u_{IC}^* - \frac{\nu}{dx^2} (u_E^* + u_W^* - 2u_{IC}^*) - \frac{\nu}{dy^2} (u_N^* + u_S^* - 2u_{IC}^*) \\ \alpha v_{IC}^* - \frac{\nu}{dx^2} (v_E^* + v_W^* - 2v_{IC}^*) - \frac{\nu}{dy^2} (v_N^* + v_S^* - 2v_{IC}^*) \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad (2.37)$$

Conventional domain boundary cells have zero thickness in the boundary normal direction. This means the cell center is located on the edge of the interior neighbor, resulting in the neighbor cell center being half the cell size away as shown in Figure 7a.

The finite difference scheme can be partitioned as shown below for the x direction.

$$\frac{\partial^2 u^*}{\partial x^2} = \frac{u_{i+1,j}^* - 2u_{i,j}^* + u_{i-1,j}^*}{dx^2} = \frac{u_{i+1,j}^* - u_{i,j}^*}{dx} - \frac{u_{i,j}^* - u_{i-1,j}^*}{dx} \quad (2.38)$$

For a west domain boundary, the second term is modified by halving the cell spacing as shown below.

$$\left. \frac{\partial^2 u^*}{\partial x^2} \right|_{Wbdry} = \frac{\frac{u_{i+1,j}^* - u_{i,j}^*}{dx} - \frac{u_{i,j}^* - u_{i-1,j}^*}{0.5dx}}{dx} = \frac{u_{i+1,j}^* - 3u_{i,j}^* + 2u_{i-1,j}^*}{dx^2} = \frac{u_E^* - 3u_{IC}^* + 2u_W^*}{dx^2} \quad (2.39)$$

The same procedure must be constructed in the y direction and combines to result in the following equations for the cells adjacent to the west domain boundary.

$$\left[\begin{array}{l} \alpha u_{IC}^* - \frac{\nu}{dx^2}(u_E^* + 2u_W^* - 3u_{IC}^*) - \frac{\nu}{dy^2}(u_N^* + u_S^* - 2u_{IC}^*) \\ \alpha v_{IC}^* - \frac{\nu}{dx^2}(v_E^* + 2v_W^* - 3v_{IC}^*) - \frac{\nu}{dy^2}(v_N^* + v_S^* - 2v_{IC}^*) \end{array} \right]_{wb} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad (2.40)$$

A visual representation of the corresponding stencil for a west conventional domain boundary conditions shown in Figure 12a. For ease of construction the operator matrix these equations can be factored in the following manner for a west boundary condition. Only the cell direction indexes of the fourth term change for other sides of the domain

$$\left[\begin{array}{l} \alpha u_{IC}^* - \frac{\nu}{dx^2}(u_E^* + u_W^* - 2u_{IC}^*) - \frac{\nu}{dy^2}(u_N^* + u_S^* - 2u_{IC}^*) - \frac{\nu}{dx^2}(u_W^* - u_{IC}^*) \\ \alpha v_{IC}^* - \frac{\nu}{dx^2}(v_E^* + v_W^* - 2v_{IC}^*) - \frac{\nu}{dy^2}(v_N^* + v_S^* - 2v_{IC}^*) - \frac{\nu}{dx^2}(v_W^* - v_{IC}^*) \end{array} \right]_{wb} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad (2.41)$$

The only difference between this and Equation 2.37 is the addition of the fourth term for the boundary adjacent cells. Since interior boundary caps are at a full cell spacing as described in Figure 7d, the fourth term is no longer needed. The west interior boundary cap stencil is shown in Figure 12b.

The boundary cell values have been already computed during application of the inlet and outlet boundary conditions. Since these velocity values are known, they must be lifted to the right hand side of the equation. First consider a cell located adjacent to a

conventional west domain boundary. After lifting the known value to the right hand side, the equation will take the following form.

$$\left[\begin{array}{l} \alpha u_{IC}^* - \frac{\nu}{dx^2}(u_E^* - 2u_{IC}^*) - \frac{\nu}{dy^2}(u_N^* + u_S^* - 2u_{IC}^*) \\ \alpha v_{IC}^* - \frac{\nu}{dx^2}(v_E^* - 2v_{IC}^*) - \frac{\nu}{dy^2}(v_N^* + v_S^* - 2v_{IC}^*) \end{array} \right]_{wb} = \left[\begin{array}{l} b_x + \frac{\nu 2u_W^*}{dx^2} \\ b_y + \frac{\nu 2v_W^*}{dx^2} \end{array} \right] \quad (2.42)$$

Since interior boundary caps allow for boundary orientations other than orthogonal, as shown in Figure 7f, the cells are often arranged in a stair step manner. When this occurs there will be interior cells which have more than one boundary cell neighbor. This is shown in Figure 7e. The equation below is the example of the boundary condition lift when the cell has a boundary cap cell to the west and a boundary cap cell to the north.

$$\left[\begin{array}{l} \alpha u_{IC}^* - \frac{\nu}{dx^2}(u_E^* - 3u_{IC}^*) - \frac{\nu}{dy^2}(u_S^* - 2u_{IC}^*) \\ \alpha v_{IC}^* - \frac{\nu}{dx^2}(v_E^* - 3v_{IC}^*) - \frac{\nu}{dy^2}(v_S^* - 2v_{IC}^*) \end{array} \right]_{W/Nb} = \left[\begin{array}{l} b_x + \frac{\nu 2u_W^*}{dx^2} + \frac{\nu u_N^*}{dy^2} \\ b_y + \frac{\nu 2v_W^*}{dx^2} + \frac{\nu v_N^*}{dy^2} \end{array} \right] \quad (2.43)$$

To enhance stability of the non-staggered grid, the pressure gradient is removed from the provisional velocity resulting in a second provisional velocity denoted by \mathbf{u}^{**} .

2.5.9 Modification to the divergence calculation

Next the divergence operation is performed on the second provisional velocity to construct the pressure Poisson equation. The term is expanded as follows for the 2 dimensional case.

$$\nabla \cdot \mathbf{u}^{**} = \sum \frac{du_i^{**}}{dx_i} = \frac{du^{**}}{dx} + \frac{dv^{**}}{dy} \quad (2.44)$$

First order central differencing can be used to approximate these derivative values. This finite difference scheme is 2nd order accurate and shown below for the x direction.

$$\nabla \cdot \mathbf{u}^{**} = \frac{u_{i+1}^{**} - u_{i-1}^{**}}{2dx} + \frac{v_{j+1}^{**} - v_{j-1}^{**}}{2dy} \quad (2.45)$$

For clarity, notations are replaced with the subscript indices of cell neighbor directions. The finite difference equation is rewritten below in terms of cell neighbors.

$$\nabla \cdot \mathbf{u}^{**} = \frac{u_E^{**} - u_W^{**}}{2dx} + \frac{v_N^{**} - v_S^{**}}{2dy} \quad (2.46)$$

This can also be expressed in terms of the values at the cell faces.

$$\nabla \cdot \mathbf{u}^{**} = \frac{\frac{u_E^{**} + u_{IC}^{**}}{2} - \frac{u_{IC}^{**} + u_W^{**}}{2}}{dx} + \frac{\frac{v_N^{**} - v_{IC}^{**}}{2} - \frac{v_{IC}^{**} - v_S^{**}}{2}}{dx} \quad (2.47)$$

If either of these faces lie on the Cartesian domain boundary, the cell spacing is not a full cell (dx) away. The finite difference scheme must be updated accordingly. This condition is illustrated in Figure 7a. Domain boundary cells are positioned on the cell face, so those values should directly be substituted into the finite difference scheme as shown below for a West domain boundary.

$$\nabla \cdot \mathbf{u}^{**} |_{Wbdry} = \frac{\frac{u_E^{**} + u_{IC}^{**}}{2} - u_W^{**}}{dx} + \frac{\frac{v_N^{**} - v_{IC}^{**}}{2} - \frac{v_{IC}^{**} - v_S^{**}}{2}}{dx} \quad (2.48)$$

For computational efficiency Equation 2.47 can be split into three arguments as shown below. The first two will be executed on all domain interior cells, and the third term will only be executed on boundary adjacent cells to account for the half cell spacing.

The equation below shows this formulation for a west boundary with only the direction indices changing for the other boundary directions.

$$\nabla \cdot \mathbf{u}^{**} = \frac{\frac{u_E^{**} + u_{IC}^{**}}{2} - \frac{u_{IC}^{**} + u_W^{**}}{2}}{dx} + \frac{\frac{v_N^{**} - v_{IC}^{**}}{2} - \frac{v_{IC}^{**} - v_S^{**}}{2}}{dx} + \frac{u_{IC}^{**} - u_W^{**}}{2} \Big|_{Wbdry} \quad (2.49)$$

Since the interior boundary cap cells are inserted into the domain boundary arrays, without modification the second term would be applied to them even though they are at a full cell spacing as illustrated in Figure 7d. Thus, the only modification to the gradient scalar calculation for interior boundary cap implementation is to remove the third term of Equation 2.49.

2.5.10 Modifications to the Pressure Poisson equation

The scalar potential φ can be calculated from the following equation which is of the Poisson type. The negative signs are added so that construction of the operator matrix is similar to that of the Helmholtz equation.

$$-\nabla^2 \varphi = -\alpha(\nabla \cdot \mathbf{u}^{**}) \quad (2.50)$$

Since this equation is a second order PDE, it is necessary to put it in the form of a system of linear equations as shown in Equation 2.30, where \mathbf{A} is the operator matrix, \mathbf{x} is the vector of unknowns, and \mathbf{b} is the solution vector. In this case the \mathbf{b} vector contains a time step scalar multiplied by the divergence of \mathbf{u}^{**} . The Poisson equation can be expanded as shown below, expressing the second derivatives of in the x and y directions respectively. For simplicity, this is demonstrated for the 2-dimensional case.

$$-\left(\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2}\right) = -\alpha(\nabla \cdot \mathbf{u}^{**}) \quad (2.51)$$

The second derivative terms can be represented by the second order accurate central differencing schemes shown below.

$$\frac{\partial^2 \varphi}{\partial x^2} = \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{dx^2} \quad (2.52)$$

$$\frac{\partial^2 \varphi}{\partial y^2} = \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{dy^2} \quad (2.53)$$

For clarity, notations defined below can replace the subscript indices with the cell neighbor direction as defined in Equation 2.7.

$$\frac{\partial^2 \varphi}{\partial x^2} = \frac{\varphi_E - 2\varphi_{IC} + \varphi_W}{dx^2} \quad (2.54)$$

$$\frac{\partial^2 \varphi}{\partial y^2} = \frac{\varphi_N - 2\varphi_{IC} + \varphi_S}{dy^2} \quad (2.55)$$

Substituting these expressions into the left side of Equation 2.51 results in the following equation. A visual representation of this stencil for interior cells is shown in Figure 13.

$$-\frac{1}{dx^2}(\varphi_E + \varphi_W - 2\varphi_{IC}) - \frac{1}{dy^2}(\varphi_N + \varphi_S - 2\varphi_{IC}) = -\alpha(\nabla \cdot \mathbf{u}^{**}) \quad (2.56)$$

When using conventional domain boundary conditions, the boundary adjacent cells must be modified for the half cell spacing. For example the finite difference term in the x direction at a west boundary can be represented by the following equation.

$$\left. \frac{\partial^2 \varphi}{\partial x^2} \right|_{Wb} = \frac{\frac{\varphi_{i+1,j} - \varphi_{i,j}}{dx} - \frac{\varphi_{i,j} - \varphi_{i-1,j}}{0.5dx}}{dx} = \frac{\varphi_{i+1,j} - 3\varphi_{i,j} + 2\varphi_{i-1,j}}{dx^2} = \frac{\varphi_E - 3\varphi_{IC} + 2\varphi_W}{dx^2} \quad (2.57)$$

The same procedure must be constructed in the y direction and combines to result in the following equation for the cells adjacent to the west domain boundary. A visual representation of this cell is shown in Figure 14a.

$$-\frac{1}{dx^2}(\varphi_E + 2\varphi_W - 3\varphi_{IC}) - \frac{1}{dy^2}(\varphi_N + \varphi_S - 2\varphi_{IC})\Big|_{wb} = -\alpha(\nabla \cdot \mathbf{u}^{**}) \quad (2.58)$$

Neumann boundary conditions are then constructed for the domain boundary.

This example will continue to consider the west boundary, again recalling $\frac{1}{2}dx$ is the cell spacing at the domain boundary.

$$\frac{\partial \varphi}{\partial n_W} = \frac{\varphi_{IC} - \varphi_W}{0.5dx} = C \quad (2.59)$$

Since φ_W is treated as a known value in the system of linear equations, an expression for this value must be created. Solving the previous equation for φ_W results in the following.

$$\varphi_W = \varphi_{IC} - 0.5Cdx \quad (2.60)$$

This can now be substituted back into Equation 2.58 to apply the boundary condition. A visual representation of the final domain adjacent cell with boundary conditions applied to the operator matrix can be seen in Figure 14a.

$$-\frac{1}{dx^2}(\varphi_E - \varphi_{IC}) - \frac{1}{dy^2}(\varphi_N + \varphi_S - 2\varphi_{IC})\Big|_{wb} = -\alpha(\nabla \cdot \mathbf{u}^{**}) - \frac{C}{dx} \quad (2.61)$$

Implementation of internal boundary caps will result with the same calculation on interior cells, but Poisson construction at the boundary caps must be modified due to all three major differences illustrated in Figure 7. The modification in Equation 2.58 for the half cell spacing at conventional domain boundary is no longer needed, thus the interior boundary cap formulation will start with Equation 2.56.

As before, a Neumann boundary condition must be constructed for the domain boundary. For interior boundary caps, the normal direction is not necessarily normal to the domain boundary as shown in Figure 7f. The Neumann boundary condition must be

constructed from the boundary cell and interpolated point between neighbors in much the same way extrapolation to the boundary was previously performed. This approach differs in the fact that the values at the neighbor cells are not known as well. Instead an analytical substitution must be derived to replace the boundary cell Poisson contribution to the operator matrix. To explain, consider a west boundary cap cell with the boundary cap pointed downward an angle less than 45 degrees. This configuration is shown in Figure 15a-b. The equation below constructs the Neumann equation using linear interpolation between the adjacent cells with the λ values representing the geometric position of the interpolation point.

$$\frac{\partial \varphi}{\partial n} = \frac{\varphi_{IC} - \varphi_P}{dn} = \frac{\lambda_1 \varphi_{IC} + \lambda_2 \varphi_S - \varphi_W}{dn} = C \quad (2.62)$$

Solving this equation for the west term and substituting into equation 2.56 will provide the following result which is shown in Figure 14b.

$$-\frac{1}{dx^2}(\varphi_E + (\lambda_1 - 2)\varphi_{IC} + \lambda_2 \varphi_S) - \frac{1}{dy^2}(\varphi_N + \varphi_S - 2\varphi_{IC}) \Big|_{wb} = -\alpha(\nabla \cdot \mathbf{u}^{**}) - \frac{c_w dn}{dx^2} \quad (2.63)$$

There exists a second interior boundary cap arrangement when substituting for the Neumann condition. The boundary adjacent cell at the stair step will have two boundary neighbors in the stencil as shown in Figure 15c. Equation 2.62 again must be applied to the boundary cell to the west. This equation is also used to create the algebraic substitution for the north boundary neighbor as well.

$$\varphi_N = \lambda_1 \varphi_{NE} + \lambda_2 \varphi_E - C_N dn \quad (2.64)$$

When solved and substituted into Equation 2.56 the following results. This stair step cell stencil is shown in Figure 14c.

$$-\frac{1}{dx^2}(\varphi_E + (\lambda_1 - 2)\varphi_{IC} + \lambda_2\varphi_S) - \frac{1}{dy^2}(\lambda_1\varphi_{NE} + \lambda_2\varphi_E + \varphi_S - 2\varphi_{IC}) \Big|_{wb} = -\alpha(\nabla \cdot \mathbf{u}^{**}) - \frac{c_W dn}{dx^2} - \frac{c_N dn}{dy^2} \quad (2.65)$$

It should be noted that the new stencil for this cell includes a north-east neighbor that wasn't in the initial stencil. This is shown in Figure 15d.

After solving the pressure Poisson equation, the values are extrapolated to the boundary as done prior. These results are used to project onto the divergence free velocity field to get the final velocity value for the current sub-iteration using the following equation where α is a scalar function of time step size.

$$\mathbf{u}^{n+1} = -\frac{\nabla\varphi}{\alpha} + \mathbf{u}^{**} \quad (2.66)$$

Finally the pressure can be updated using the equation below

$$p^{n+1} = \varphi - \frac{1}{Re}(\nabla \cdot \mathbf{u}^*) \quad (2.67)$$

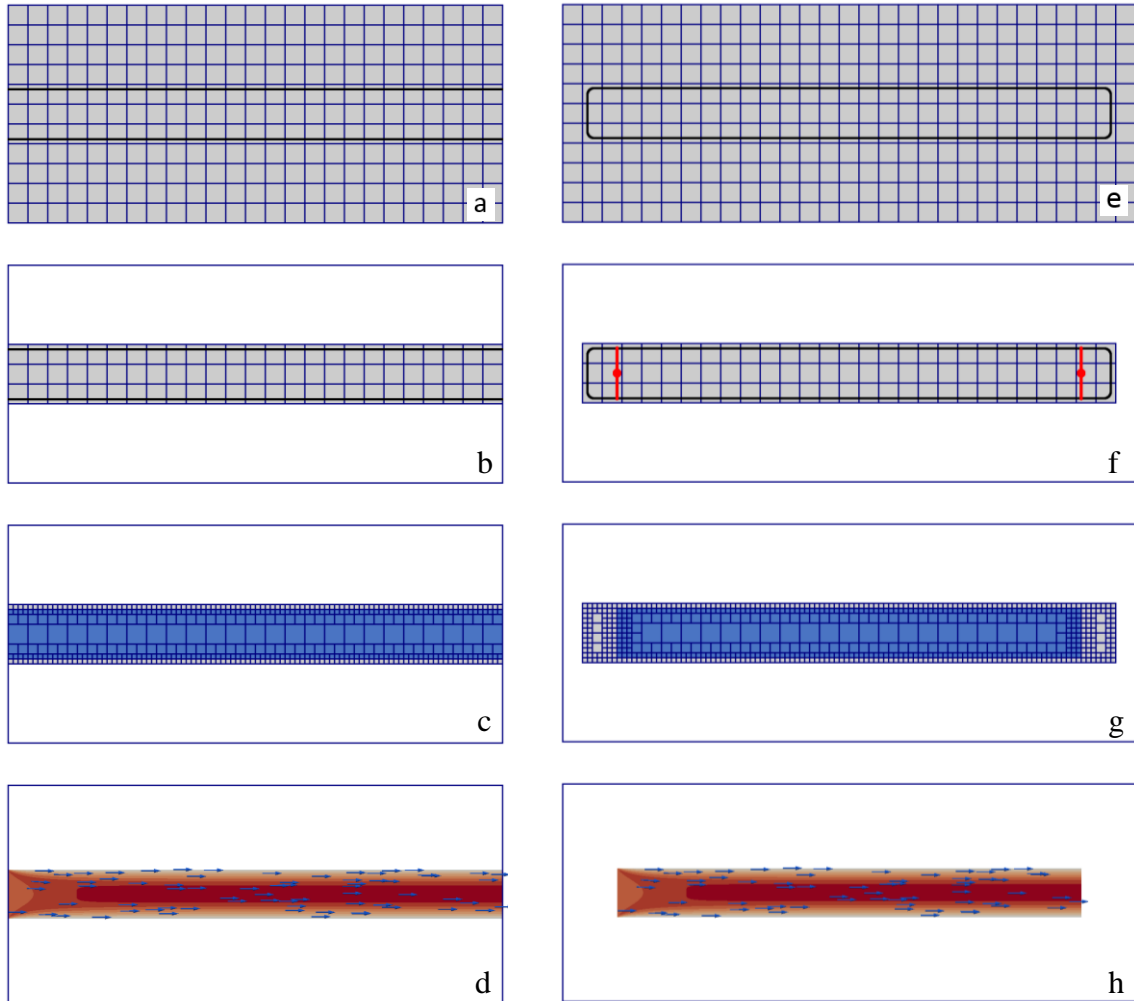


Figure 4: Comparison of computational domain construction between (a-d) conventional domain boundary conditions and (e-h) interior boundary caps. (a,e) set up initial grid and insert geometry, (b,f) prune mesh and apply boundary caps, (c,g) refine mesh and classify cells as fluid/solid, (d,h) perform flow simulation.

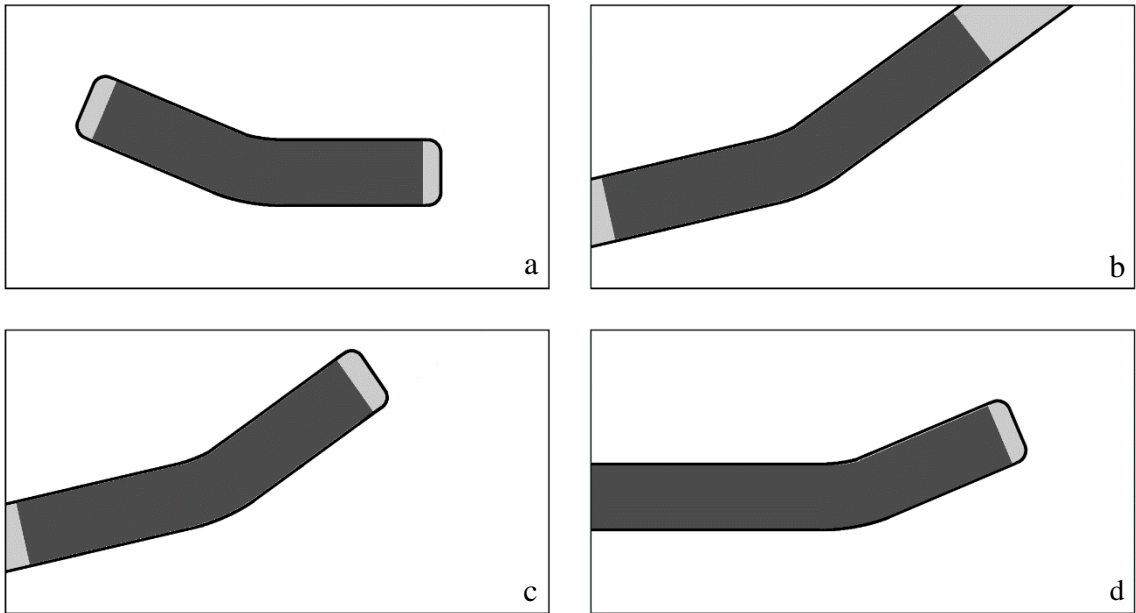


Figure 5: Some of the new geometric configurations interior boundary caps allow. The fluid domain is marked in dark gray. (a) Entire geometry inside the domain, (b) All inlets/outlets crossing domain boundaries, (c) Combination of each, (d) Interior boundary caps used with conventional domain boundary conditions.

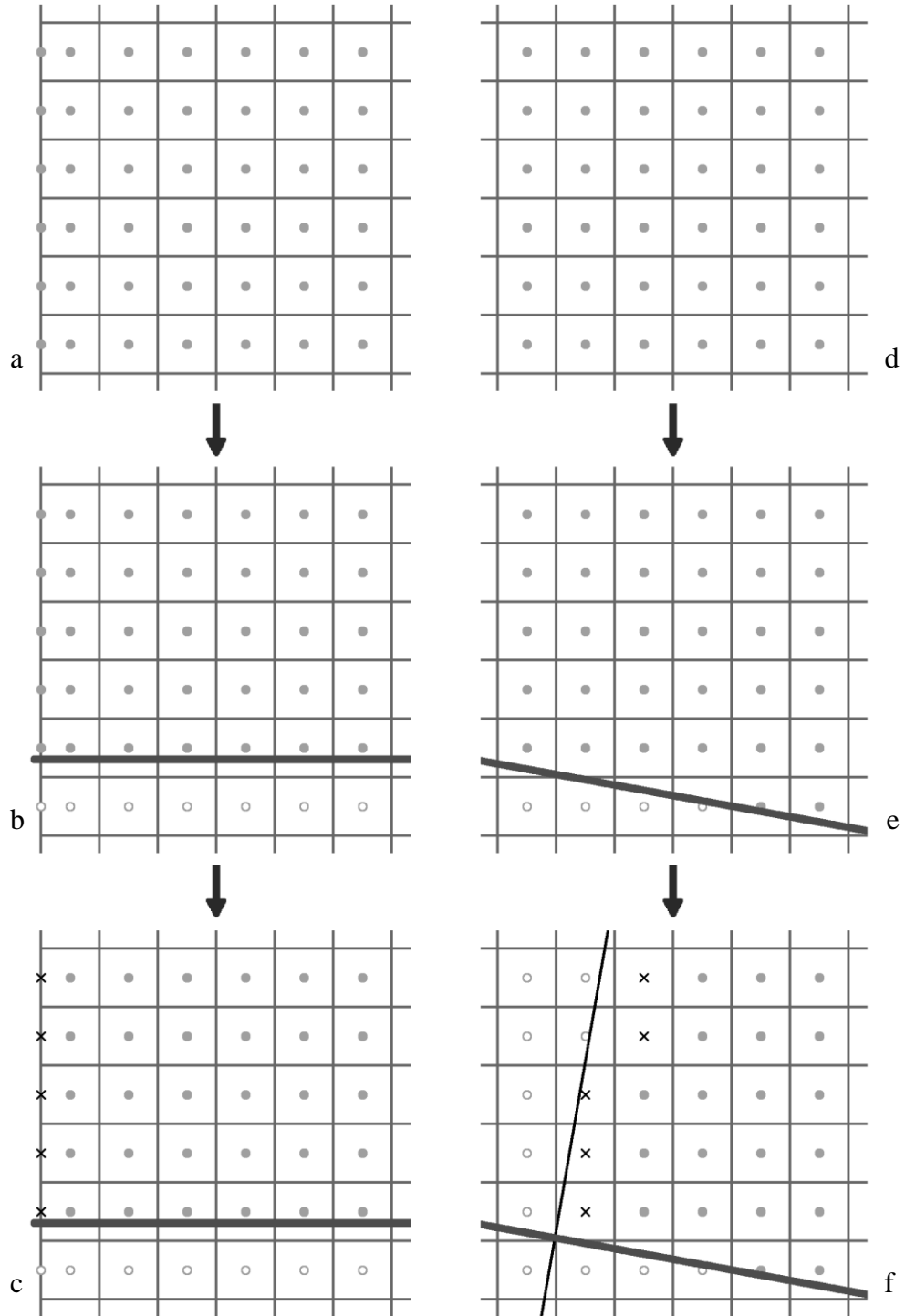


Figure 6: Comparison of how boundary conditions cells are selected (a-c) conventional domain boundaries (d-f) interior boundary conditions, (a,d) initial grid, (b,e) insert geometry and reclassify outside cells as solid, (c) level set values are used to identify domain boundary cells, (f) level set values and boundary cap plane are used to identify boundary cap cells and cells outside the place are changed to solid.

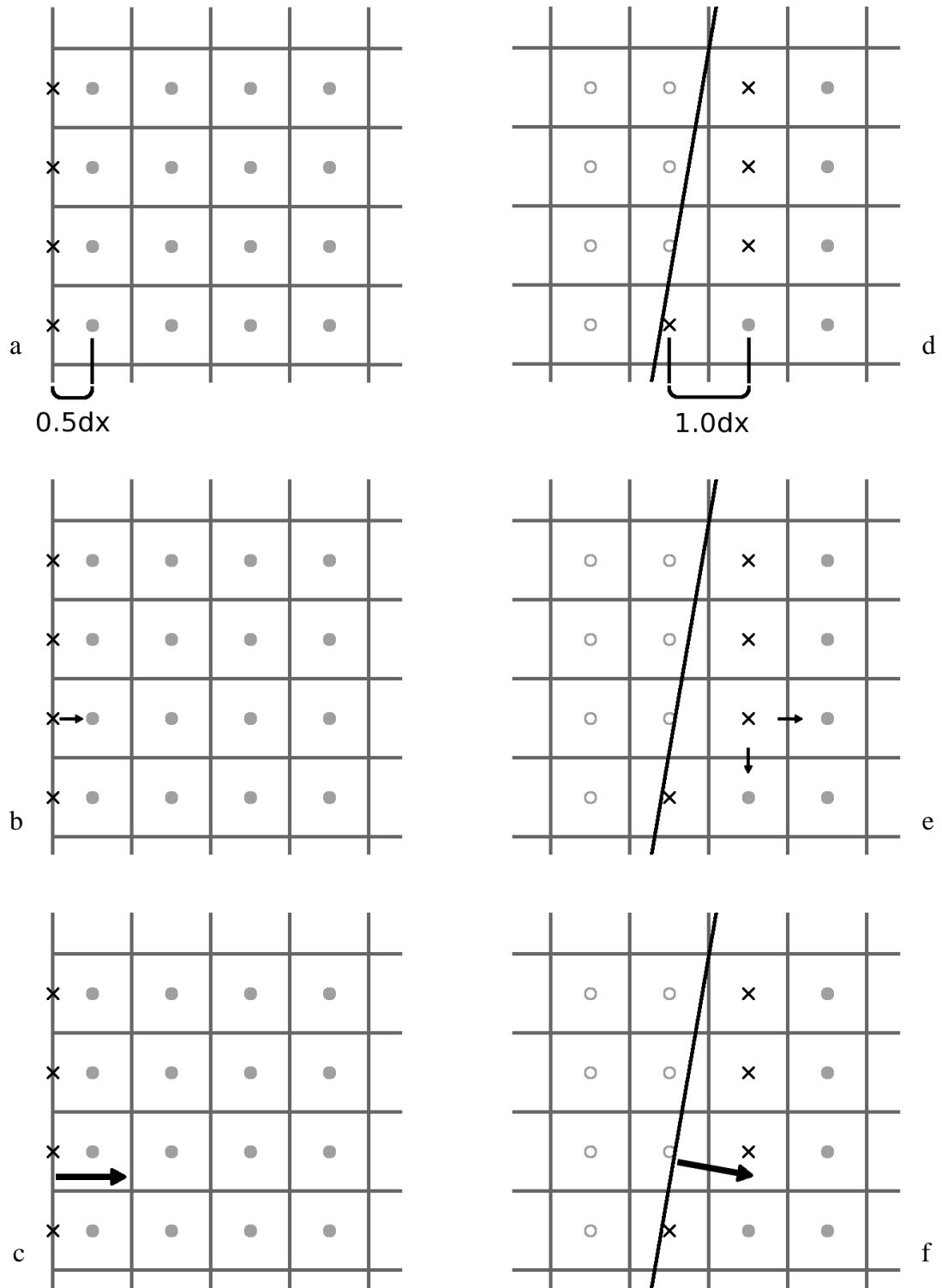


Figure 7: Differences between conventional domain boundary cells (a-c) and interior boundary cap cells (d-f). (a,d) boundary cell spacing, (b,e) flux contribution to neighbors, (c,f) boundary condition normal.

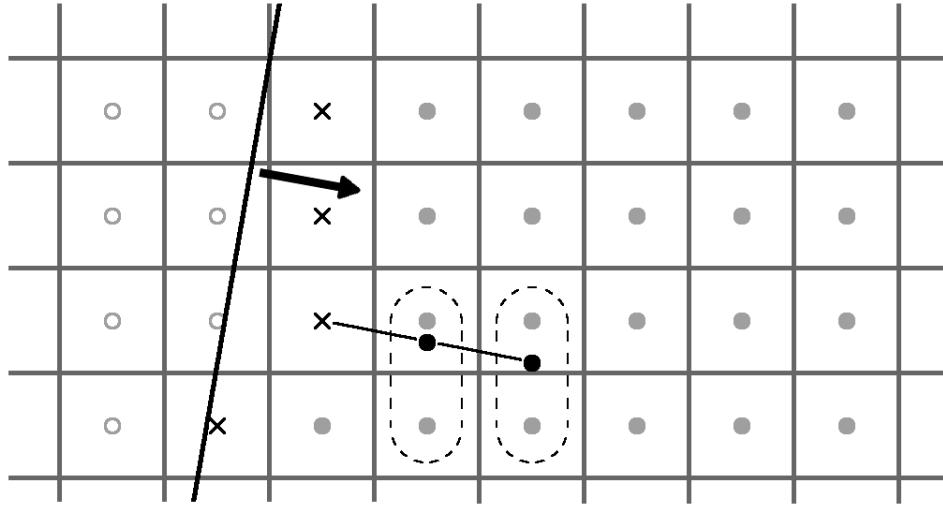


Figure 8: Linear extrapolation to boundary. Interpolation between the encircled cells is used to determine values of the probe points (solid black circles).

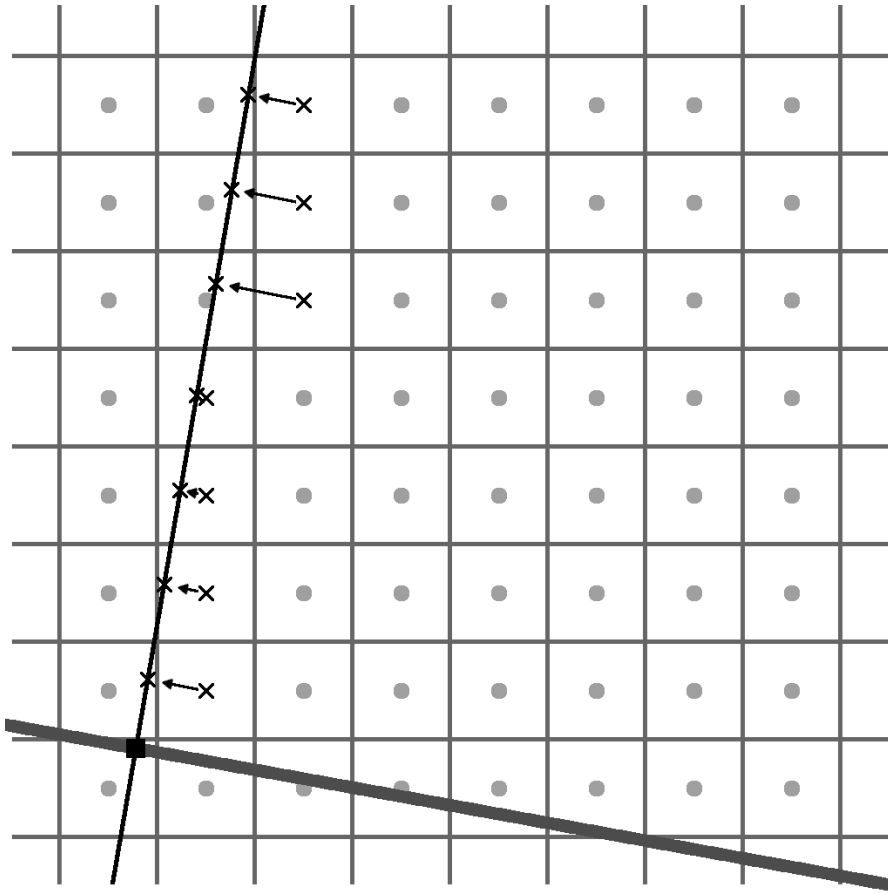


Figure 9: Dirichlet boundary condition projection for interior boundary conditions.

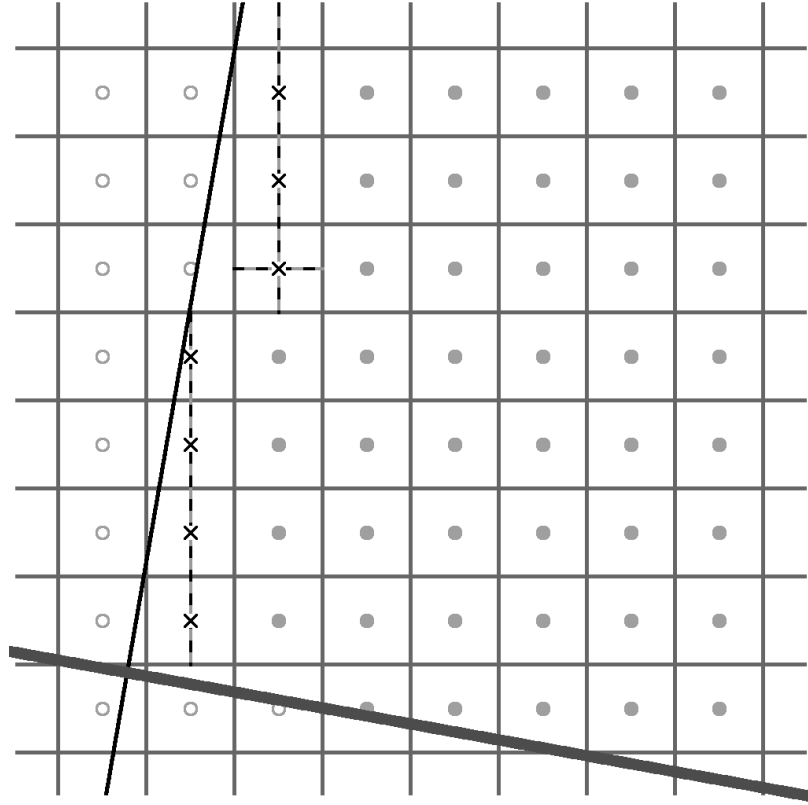


Figure 10: Calculation of mass flux using stair step method.

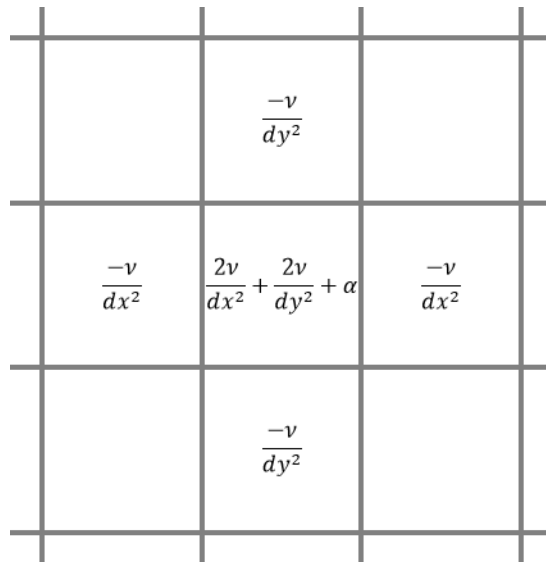


Figure 11: Helmholtz equation stencil of coefficients for interior cells.

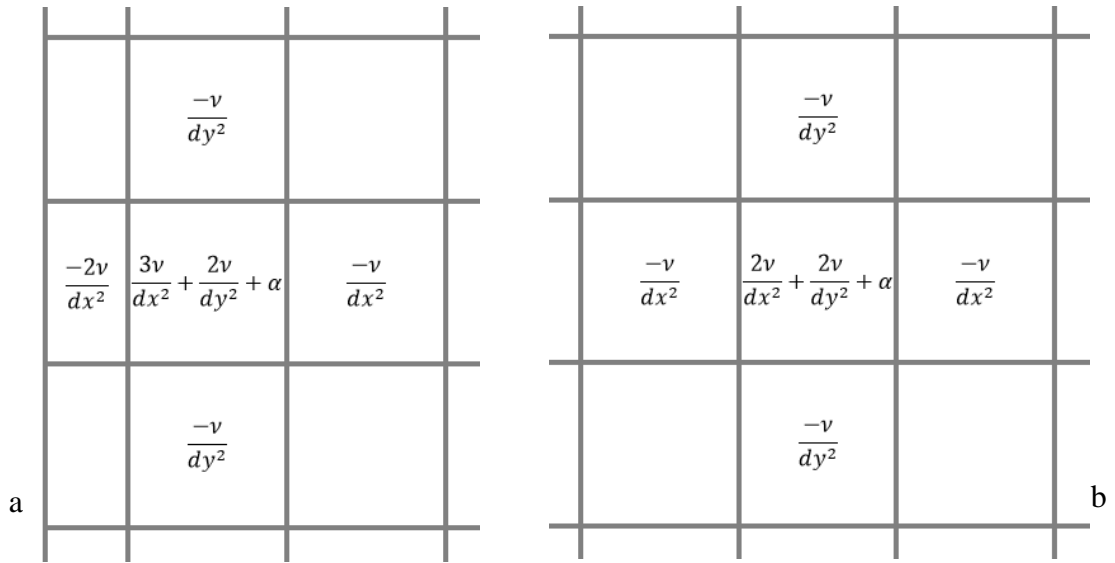


Figure 12: Helmholtz equation stencil of coefficients for boundary adjacent cells (a) conventional domain boundaries (b) interior boundary conditions.

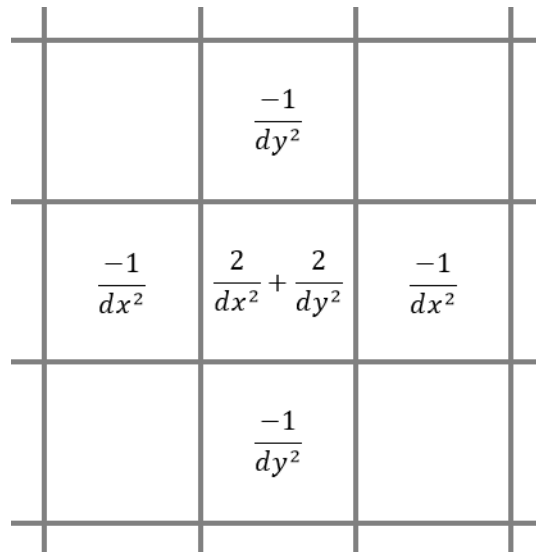


Figure 13: Pressure Poisson equation stencil of coefficients for interior cells.

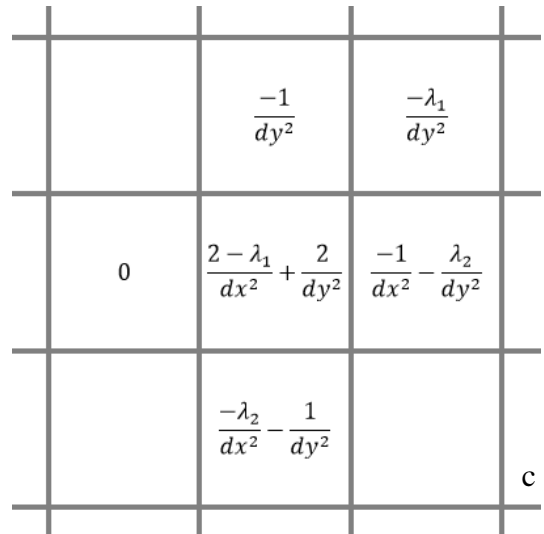
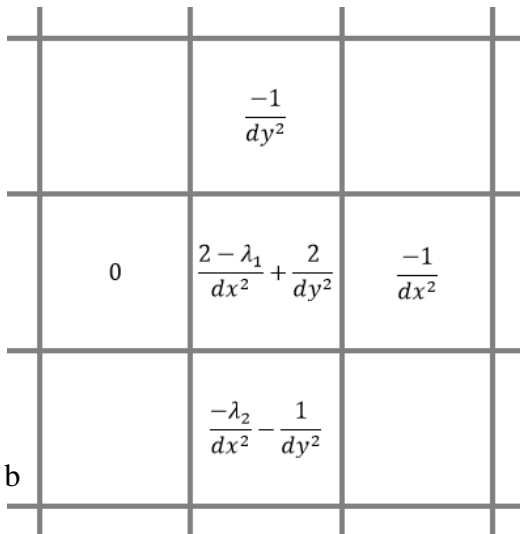
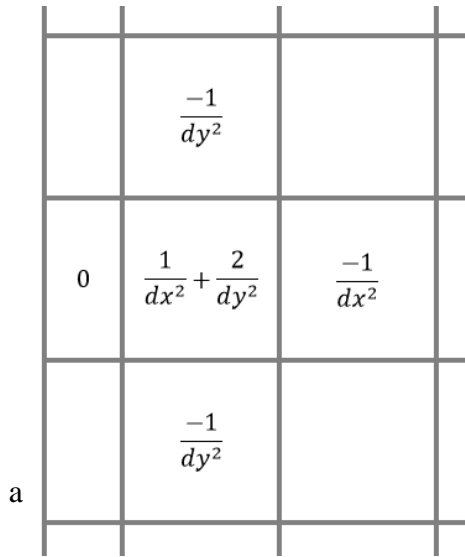


Figure 14: Pressure Poisson equation stencil of coefficients for west boundary adjacent cells (a) conventional domain boundaries, (b) typical interior boundary conditions, (c) stair step interior boundary conditions.

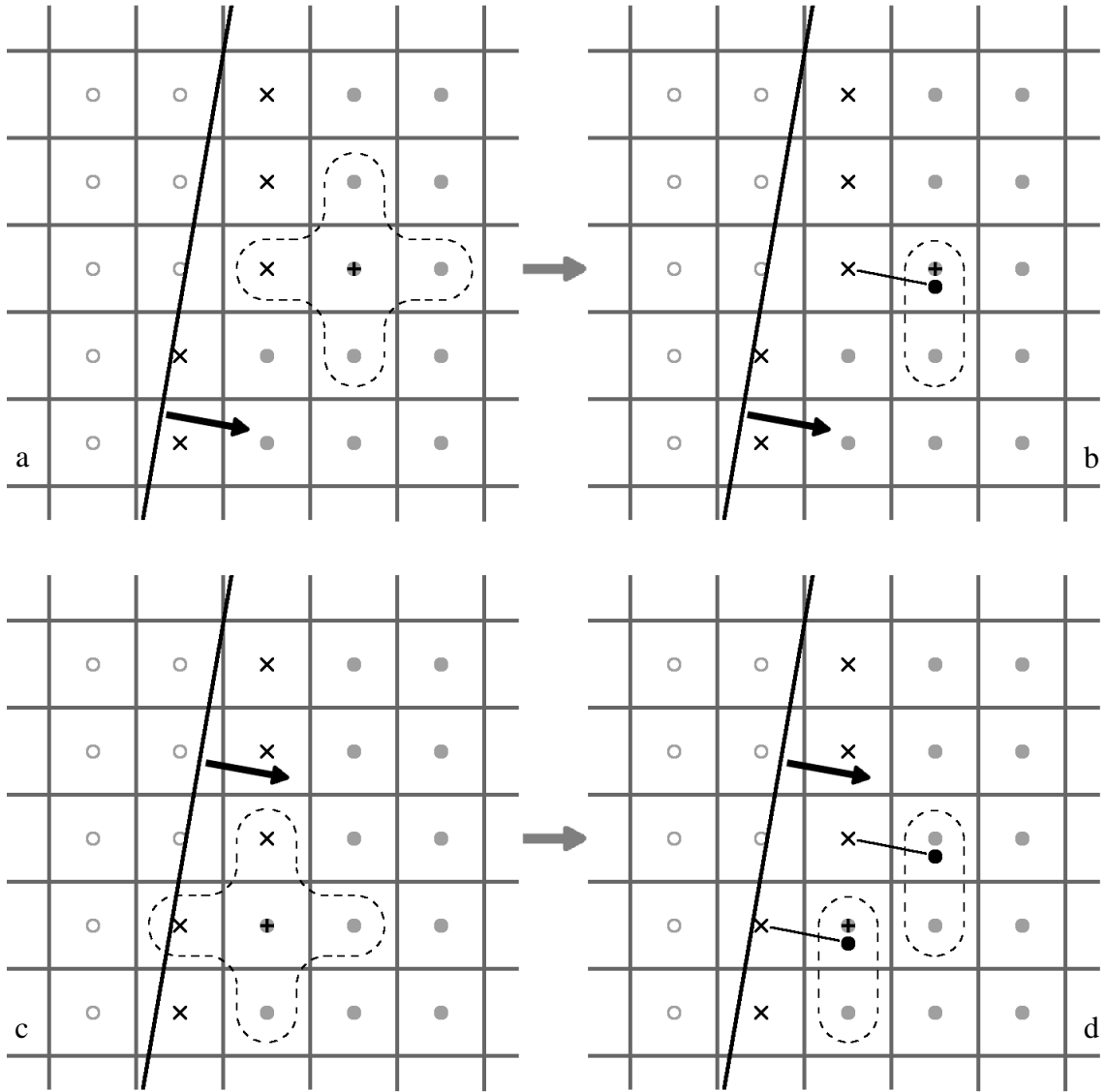


Figure 15: Neumann conditions for construction of the Poisson operator matrix at boundary adjacent cells noted by a plus, (a) typical boundary adjacent stencil including a single boundary neighbor, (b) indication of cells needed to apply substitution for boundary cell, (c) boundary adjacent stencil at stair step showing two boundary neighbors, (d) boundary adjacent stencil for stair step cell for substitution of boundary cells.

CHAPTER 3 VALIDATION

Upon completing all of the modifications listed above, it is important to validate the successful implementation of the interior boundary cap method. It is necessary to compare the boundary cap results with analytical results, simulations performed using conventional domain boundary conditions, and simulations performed using commercial software. It should be noted that all modified finite difference schemes have maintained the individual order of accuracy.

3.1 Validation of Poiseuille flow

Poiseuille flow is the first case considered to validate the implementation of interior boundary caps. An interior boundary cap simulation at 20 degree decline is compared to a conventional domain simulation. Both have a plug inlet velocity assigned to the left end ($Re = 100$). Figure 16 shows the resulting velocity magnitude contours which are in excellent agreement.

Velocity profiles of these two cases are also compared in Figure 17. Figure 17a shows the center line velocity profile along the length of the tube for both simulations. Not only do these profiles match very well, but they also show the length of development is in good agreement with the analytical solution of 6 diameters, calculated using the following equation for laminar flow.

$$L_e = 0.06(Re)D \quad (3.1)$$

Figure 17b shows the velocity profiles at a cross section located at 8 diameter lengths from the inlet. There is good agreement between the velocity profiles. The

maximum centerline velocity at this fully developed section is also in excellent agreement with the analytical solution (Table 1). Some of this error can be attributed to the centerline of the tube being at the coarsest level of mesh refinement.

3.2 Validation of 2D aortic arch with commercial software

The next validation piece is selected to demonstrate the full capability possible with the implementation of interior boundary caps. A 2D idealized aorta model is simulated using interior boundary caps and with commercial software ANSYS® Fluent. A body-fitted surface mesh was generated for the Fluent simulation. The interior boundary cap simulation utilized a total of five boundary caps. Three of these were mapped to the north domain boundary and two being mapped to the south boundary. The angle of the boundary cap ranges from horizontal (0 degrees) to 41 degrees. A plug inlet velocity ($Re = 100$) is applied to the ascending aorta, with the four outlets being constrained by mass split. Figure 18 shows velocity magnitude contours for both software packages. The results show excellent agreement. To further compare these results, velocity profiles were extracted for comparison at the ascending aorta, between the second and third branching vessels, and in the descending aorta. The results are in excellent agreement as shown below in Figure 19.

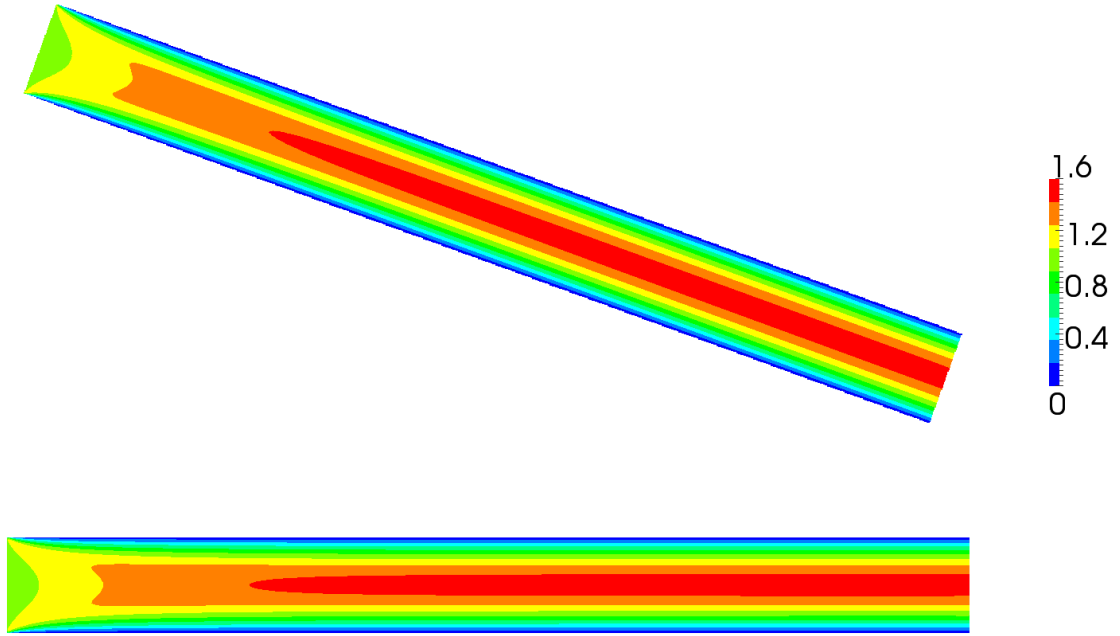


Figure 16: Velocity magnitude comparison ($Re = 100$) using pELAFINT3D with interior boundary caps (top) to pELAFINT3D using conventional domain boundaries (bottom).

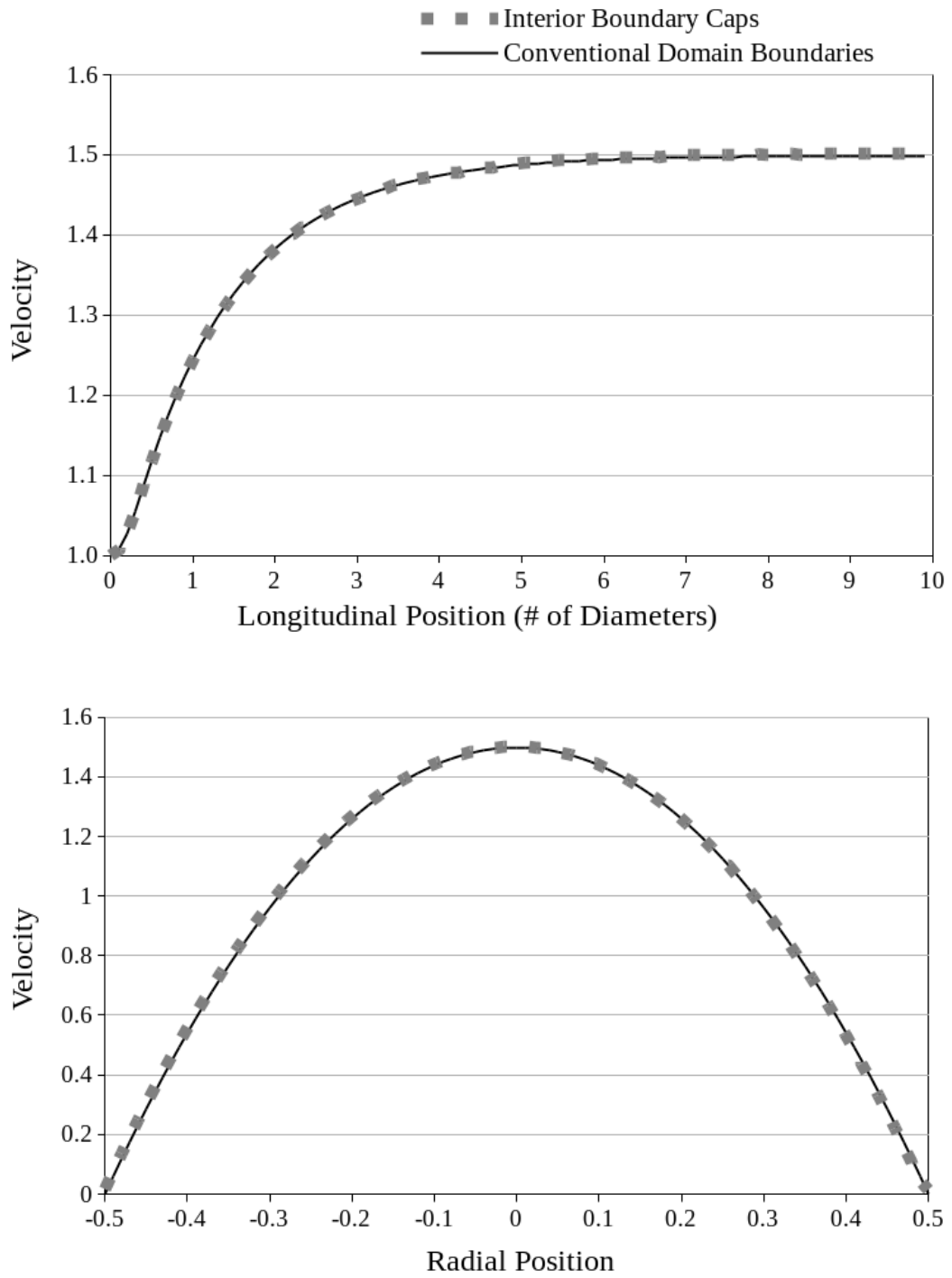


Figure 17: Poiseuille flow validation at $Re = 100$. (top) longitudinal centerline velocity, (bottom) fully developed radial velocity profile at 8 diameters from the inlet.

Table 1: Comparison of boundary cap maximum centerline velocity to the analytical value

Boundary Cap	Analytical	Error
1.502	1.500	0.001

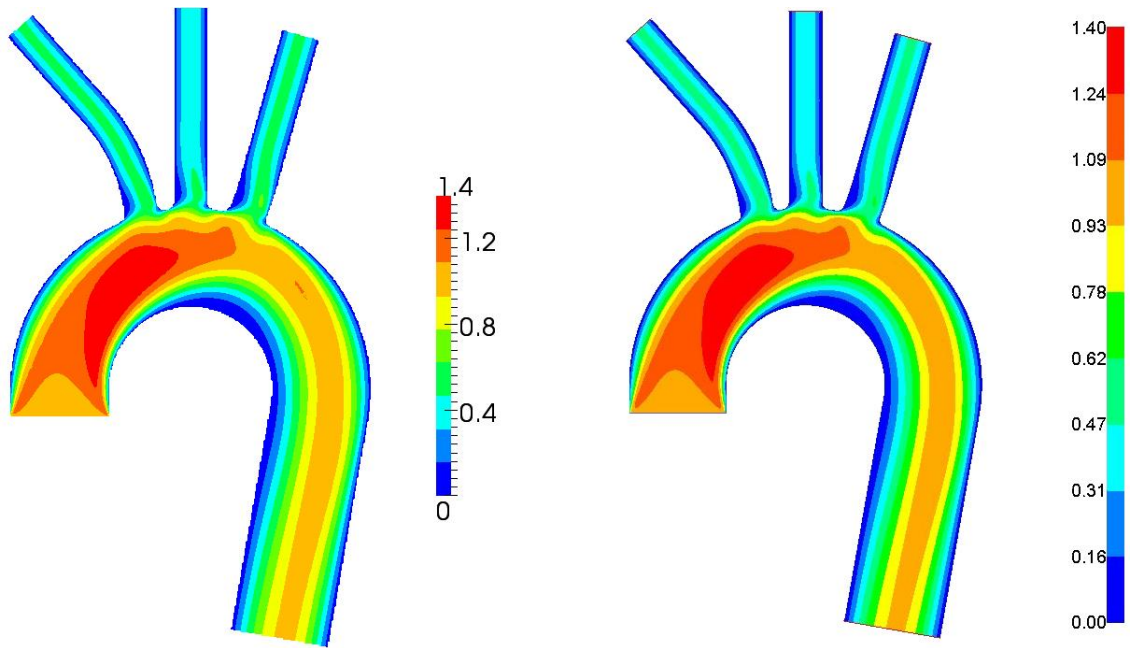


Figure 18: Velocity magnitude comparison ($Re = 100$) between pELAFINT3D using interior boundary caps (left) and ANSYS® Fluent (right).

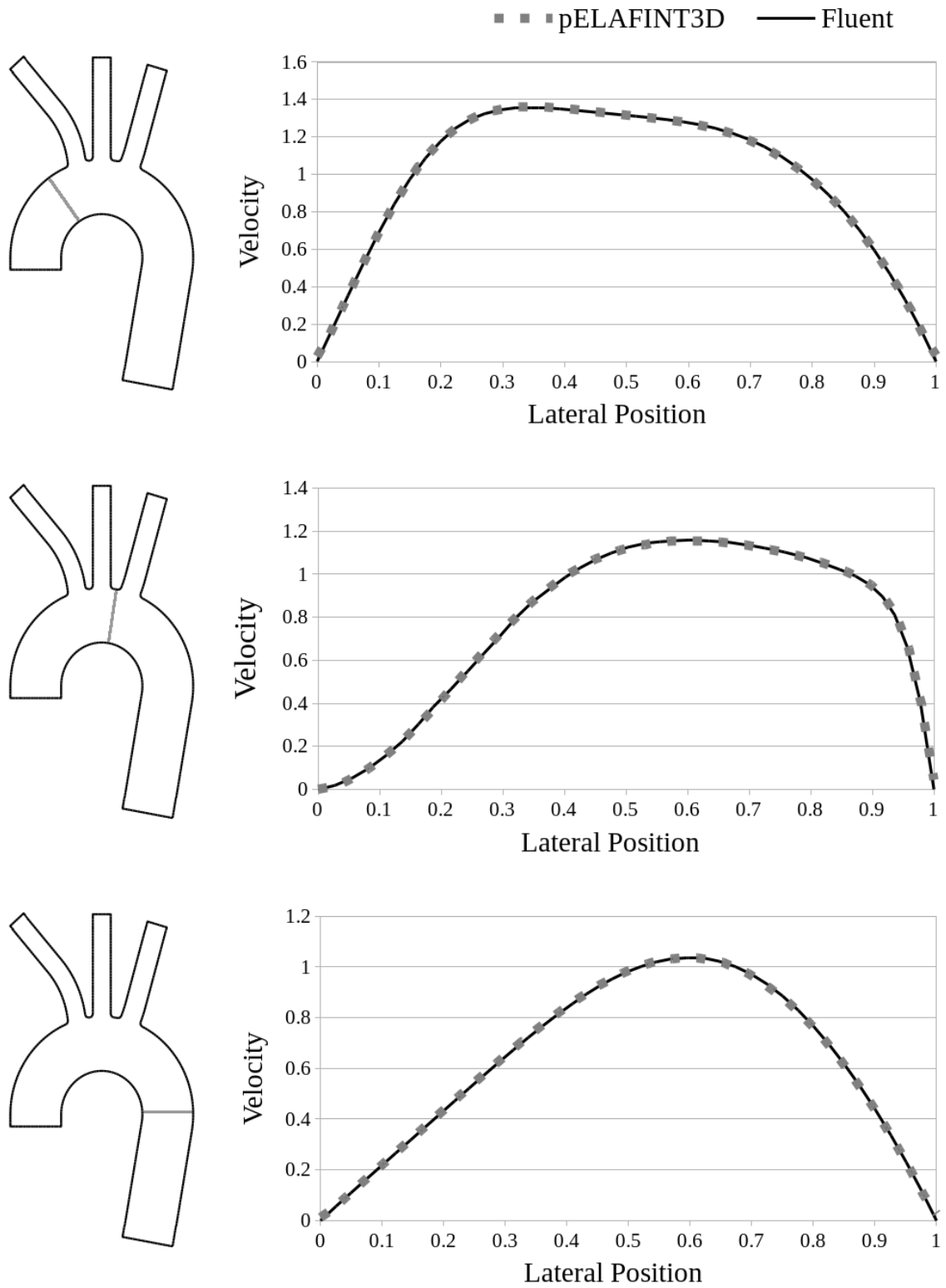


Figure 19: Comparison of velocity profiles between pELAFINT3D with interior boundary caps and commercial software Fluent at selected locations.

CHAPTER 4 EMPLOYING INTERIOR BOUNDARY CAPS TO MODEL IMAGE-BASED GEOMETRY

4.1 Image-to-flow on patient-specific data

The method of interior boundary caps can now be extended to patient-specific images. Figure 20a shows the image data of a patient's descending aorta. This is often a region of interest, as abdominal aortic aneurysms are known to occur in this region. An active contours method of image segmentation was used to generate the level set field shown in Figure 20b [27]. Interior boundary caps were then applied to the level set allowing for flow simulation to be completed. The resulting velocity magnitude contours are shown in Figure 20c. This simulation was performed at steady state with a Reynolds number of 115 achieved by supplying a parabolic inlet velocity profile. Neither flow extensions nor a surface mesh were used to conduct this simulation. This example clearly shows the benefit of interior boundary caps.

4.2 Other image-to-flow uses

While the method of interior boundary caps is an important step toward patient specific modeling in the clinical setting, it can also be useful for many other applications. Interior boundary caps decrease the amount of skill, training, and third party software needed for educational purposes or experimental investigations.

4.2.1 Image-to-flow on computer drawn image

For example, one can now create an image using a computer drawing program, identify the location of the interior boundary caps, and perform a flow simulation. Neither a surface mesh nor flow extensions are needed. An example of this is shown in

Figure 21. The computer drawn image of 128 x 128 pixels is shown on the left with the resulting velocity contours shown on the right. A parabolic velocity was applied to the inlet at the upper left at a Reynolds number of 100.

4.2.2 Image-to-flow on hand drawn image

If generation of an image by computer drawing program is burdensome, the image can simply be drawn by hand as shown below in Figure 22a. This image loosely represents a fusiform aneurysm. The image was then scanned and converted to a grayscale image. A parabolic inlet velocity was applied to the left end at a Reynolds number of 100. The results of the flow simulation is shown in Figure 22b.

A second example of a hand drawn image is shown in Figure 23a. Rather than using a scanner, a cellphone camera captured the image. It again was converted to grayscale to conduct the flow simulation. The results are shown in Figure 23b with a parabolic inlet velocity applied to the left at a Reynolds number of 100.

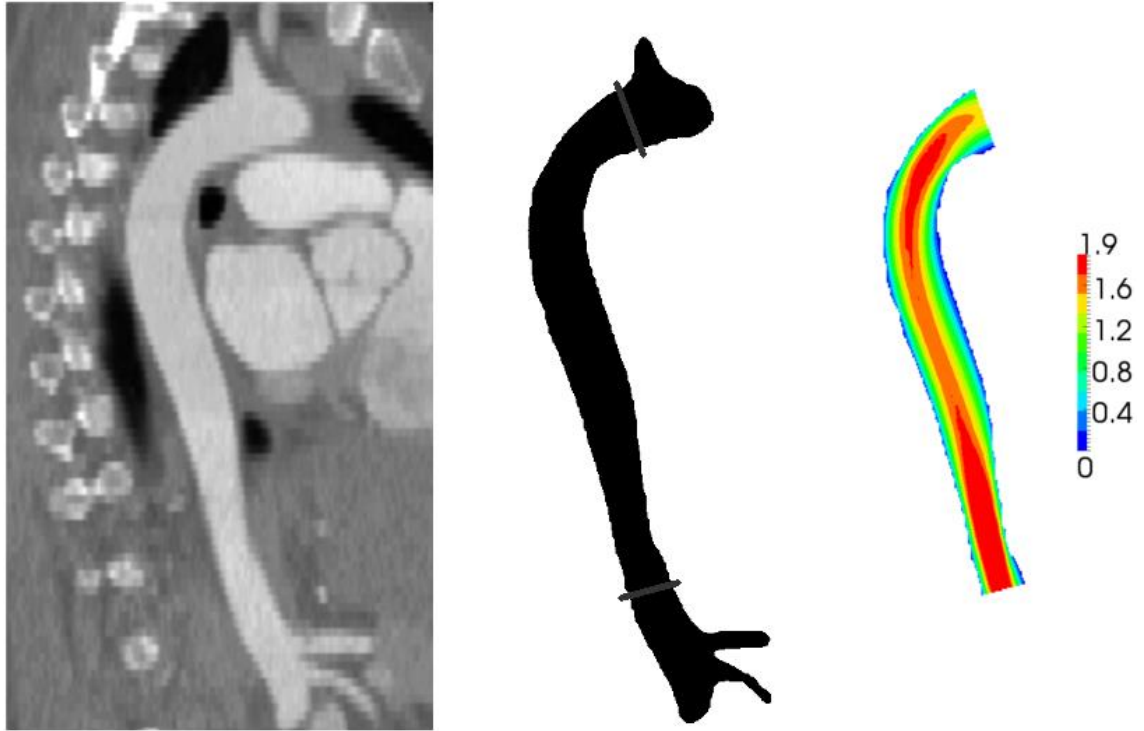


Figure 20: Image to flow simulation on patient specific data. (left) patient image data of descending aorta, (center) level set field generated from image segmentation with interior boundary cap locations marked, (right) velocity magnitude contours at Re of 115 with parabolic inlet velocity profile.

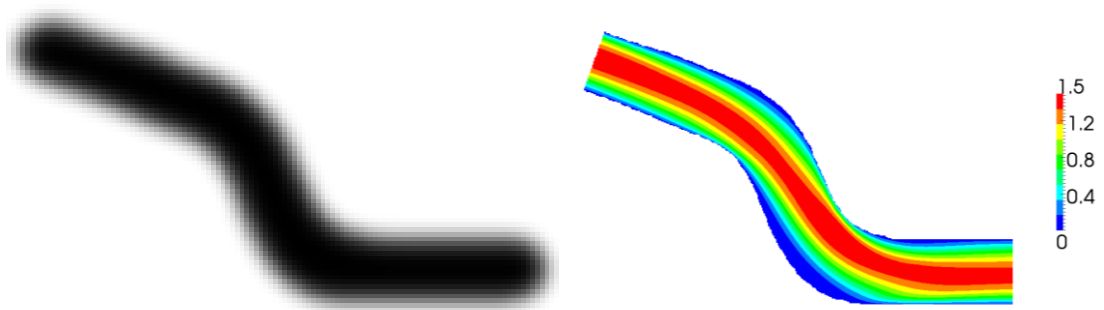


Figure 21: Image-to-flow on computer drawn image. (left) 128 x 128 pixel image, (right) Velocity magnitude contours at $Re = 100$.

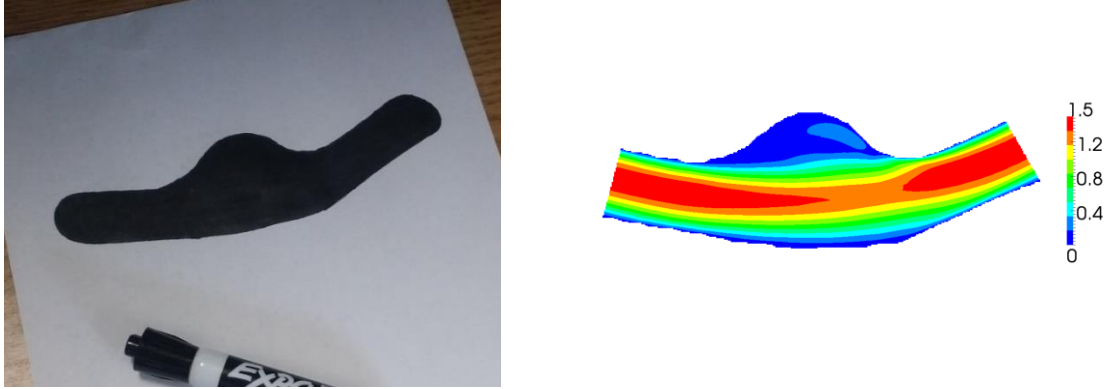


Figure 22: Image-to-flow via scanner. (left) hand drawn image, (right) Velocity magnitude contours at $Re = 100$.

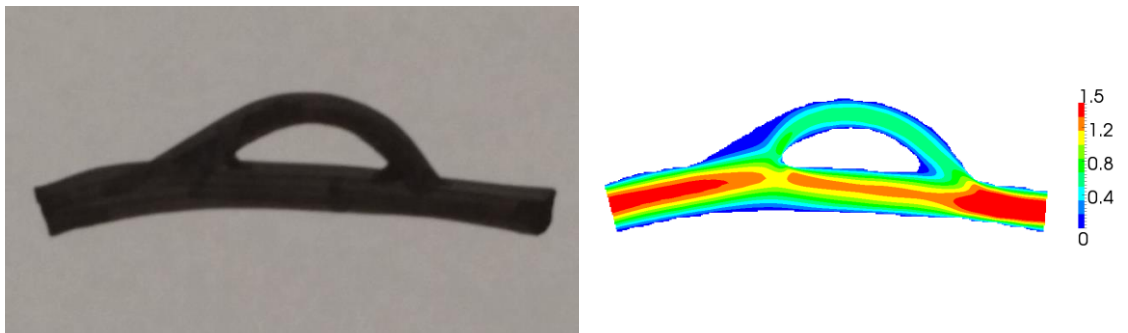


Figure 23: Image-to-flow via cellphone camera. (left) hand drawn image, (right) Velocity magnitude contours at $Re = 100$.

CHAPTER 5 CONCLUSIONS

5.1 Summary

The method of interior boundary cap has been developed, validated, and demonstrated to be an alternative to using Cartesian grid solvers with flow extensions. Interior boundary caps eliminate the burdensome constraints of vessel extension and orientation to the sides of the cuboidal domain boundary, allowing for the flexibility of Cartesian grid solvers to accommodate the wide variety of tortuous and maligned patient-specific geometries. This is a necessity before clinical applications of patient-specific modeling are to be implemented with a Cartesian solver. While this method is a significant step toward patient-specific modeling in the clinical environment, there exists opportunity to further automate the modeling process and provide clinicians with tools to make predictive analysis of disease progressions and treatment strategies.

5.2 Limitations and future work

While the method of interior boundary caps is an exciting development in the drive for patient specific modeling in the clinical setting, it is currently only developed for 2D applications. Extension to 3D should be fairly straightforward. While the discretization scheme modifications to the flow solver will be similar, there are other challenges that must be overcome. For example, the geometric stencils for constructing the directional derivatives at the boundary will become increasingly complex in three dimensions and thus implementing a least squares approach would be a suitable alternative. Also the calculation of boundary cap area and mass flux in three dimensions will be more complicated. Alternative methods such as cut cells should be considered.

One of the biggest strengths of Cartesian grid solvers is their ability to accommodate large displacements and moving objects. Currently, the method of internal boundary caps does not accommodate displacement of the vessel. Coronary artery simulation is one example in which this capability could be of use. It has been shown that the displacement of the coronary vessels throughout the cardiac cycle can have significant impacts on values of wall shear stress and oscillatory shear index [29]. A second type of vessel motion should also be considered. The current implementation of interior boundary caps does not accommodate vessel compliance. Vessel wall displacement has been shown to significantly impact patient simulation results as well [30], [31].

Extension to 3D, vessel displacement, and vessel wall elasticity in that order of development will continue to increase the usefulness and capability of patient-specific CFD models in the clinical setting.

REFERENCES

- [1] H. C. Groen, F. J. H. Gijssen, A. van der Lugt, M. S. Ferguson, T. S. Hatsukami, A. F. W. van der Steen, C. Yuan, and J. J. Wentzel, “Plaque Rupture in the Carotid Artery Is Localized at the High Shear Stress Region A Case Report,” *Stroke*, vol. 38, no. 8, pp. 2379–2381, Aug. 2007.
- [2] M. Esmaily-Moghadam, T.-Y. Hsia, and A. L. Marsden, “The assisted bidirectional Glenn: A novel surgical approach for first-stage single-ventricle heart palliation,” *J. Thorac. Cardiovasc. Surg.*, vol. 149, no. 3, pp. 699–705, Mar. 2015.
- [3] I. Borazjani, L. Ge, and F. Sotiropoulos, “High-Resolution Fluid–Structure Interaction Simulations of Flow Through a Bi-Leaflet Mechanical Heart Valve in an Anatomic Aorta,” *Ann. Biomed. Eng.*, vol. 38, no. 2, pp. 326–344, Oct. 2009.
- [4] C. A. Taylor, T. A. Fonte, and J. K. Min, “Computational Fluid Dynamics Applied to Cardiac Computed Tomography for Noninvasive Quantification of Fractional Flow Reserve Scientific Basis,” *J. Am. Coll. Cardiol.*, vol. 61, no. 22, pp. 2233–2241, Jun. 2013.
- [5] N. H. J. Pijls, B. de Bruyne, K. Peels, P. H. van der Voort, H. J. R. M. Bonnier, J. Bartunek, and J. J. Koolen, “Measurement of Fractional Flow Reserve to Assess the Functional Severity of Coronary-Artery Stenoses,” *N. Engl. J. Med.*, vol. 334, no. 26, pp. 1703–1708, Jun. 1996.
- [6] G. Xiong, G. Choi, and C. A. Taylor, “Virtual interventions for image-based blood flow computation,” *Comput.-Aided Des.*, vol. 44, no. 1, pp. 3–14, Jan. 2012.
- [7] J. R. Cebral, F. Mut, D. Sforza, R. Löhner, E. Scriverano, P. Lylyk, and C. Putman, “Clinical application of image-based CFD for cerebral aneurysms,” *Int. J. Numer. Methods Biomed. Eng.*, vol. 27, pp. 977–992, 2010.
- [8] C. A. Taylor and D. A. Steinman, “Image-based modeling of blood flow and vessel wall dynamics: applications, methods and future directions,” *Ann. Biomed. Eng.*, vol. 38, pp. 1188–1203, 2010.
- [9] E. J. Bekkers and C. A. Taylor, “Multiscale Vascular Surface Model Generation From Medical Imaging Data Using Hierarchical Features,” *IEEE Trans. Med. Imaging*, vol. 27, no. 3, pp. 331–341, Mar. 2008.
- [10] J. A. Sethian, *Level-Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge: Cambridge University Press, 1999.
- [11] S. Osher, “Fronts Propagating with Curvature Dependent Speed Algorithms Based on Hamilton-Jacobi Formulations,” *J. Comput. Phys.*, vol. 79, pp. 12–49, 1988.

- [12] W. E. Lorensen and H. E. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm,” in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1987, pp. 163–169.
- [13] K. Yokoi, F. Xiao, H. Liu, and K. Fukasaku, “Three-dimensional numerical simulation of flows with complex geometries in a regular Cartesian grid and its application to blood flow in cerebral artery with multiple aneurysms,” *J. Comput. Phys.*, vol. 202, no. 1, pp. 1–19, Jan. 2005.
- [14] T. Deschamps, P. Schwartz, D. Trebotich, P. Colella, D. Saloner, and R. Malladi, “Vessel segmentation and blood flow simulation using Level-Sets and Embedded Boundary methods,” *Int. Congr. Ser.*, vol. 1268, pp. 75–80, Jun. 2004.
- [15] K. Kumahata, M. Watanabe, and T. Matsuzawa, “Blood Flow Simulation System with Interaction between Blood Flow and Blood Vessel Wall using Image Based Cartesian Grid,” *J. Biomech. Sci. Eng.*, vol. 3, no. 2, pp. 85–100, 2008.
- [16] M. Garbey and B. Hadri, “Toward a Real Time, Image Based CFD,” in *Domain Decomposition Methods in Science and Engineering XVII*, U. Langer, M. Discacciati, D. E. Keyes, O. B. Widlund, and W. Zulehner, Eds. Springer Berlin Heidelberg, 2008, pp. 509–515.
- [17] R. Botnar, E. Nagel, M. B. Scheidegger, E. M. Pedersen, O. Hess, and P. Boesiger, “Assessment of prosthetic aortic valve performance by magnetic resonance velocity imaging,” *Magn. Reson. Mater. Phys. Biol. Med.*, vol. 10, no. 1, pp. 18–26, Feb. 2000.
- [18] S. Marella, S. Krishnan, H. Liu, and H. S. Udaykumar, “Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations,” *J. Comput. Phys.*, vol. 210, no. 1, pp. 1–31, Nov. 2005.
- [19] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, “A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method),” *J. Comput. Phys.*, vol. 152, no. 2, pp. 457–492, Jul. 1999.
- [20] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, and A. von Loebbecke, “A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries,” *J. Comput. Phys.*, vol. 227, no. 10, pp. 4825–4852, May 2008.
- [21] Y.-H. Tseng and J. H. Ferziger, “A ghost-cell immersed boundary method for flow in complex geometry,” *J. Comput. Phys.*, vol. 192, no. 2, pp. 593–623, Dec. 2003.
- [22] H. Choi and P. Moin, “Effects of the Computational Time Step on Numerical Solutions of Turbulent Flow,” *J. Comput. Phys.*, vol. 113, no. 1, pp. 1–4, Jul. 1994.

- [23] J. Yang and F. Stern, “Sharp interface immersed-boundary/level-set method for wave–body interactions,” *J. Comput. Phys.*, vol. 228, no. 17, pp. 6590–6616, Sep. 2009.
- [24] J. Yang and F. Stern, “A simple and efficient direct forcing immersed boundary framework for fluid–structure interactions,” *J. Comput. Phys.*, vol. 231, no. 15, pp. 5029–5061, Jun. 2012.
- [25] J. Mousel, “A massively parallel adaptive sharp interface solver with application to mechanical heart valve simulations,” *Theses Diss.*, Jan. 2012.
- [26] T. Tu, D. R. O’Hallaron, and O. Ghattas, “Scalable Parallel Octree Meshing for TeraScale Applications,” in *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, 2005*, pp. 4–4.
- [27] S. I. Dillard, J. A. Mousel, L. Shrestha, M. L. Raghavan, and S. C. Vigmostad, “From medical images to flow computations without user-generated meshes,” *Int. J. Numer. Methods Biomed. Eng.*, vol. 30, no. 10, pp. 1057–1083, Oct. 2014.
- [28] H. Luo, H. Dai, P. J. S. A. Ferreira de Sousa, and B. Yin, “On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries,” *Comput. Fluids*, vol. 56, pp. 61–76, Mar. 2012.
- [29] R. Torii, J. Keegan, N. B. Wood, A. W. Dowsey, A. D. Hughes, G.-Z. Yang, D. N. Firmin, S. A. M. Thom, and X. Y. Xu, “MR Image-Based Geometric and Hemodynamic Investigation of the Right Coronary Artery with Dynamic Vessel Motion,” *Ann. Biomed. Eng.*, vol. 38, no. 8, pp. 2606–2620, Apr. 2010.
- [30] N. M. Maurits, G. E. Loots, and A. E. P. Veldman, “The influence of vessel wall elasticity and peripheral resistance on the carotid artery flow wave form: A CFD model compared to in vivo ultrasound measurements,” *J. Biomech.*, vol. 40, no. 2, pp. 427–436, 2007.
- [31] F. Kabinejadian and D. N. Ghista, “Compliant model of a coupled sequential coronary arterial bypass graft: Effects of vessel wall elasticity and non-Newtonian rheology on blood flow regime and hemodynamic parameters distribution,” *Med. Eng. Phys.*, vol. 34, no. 7, pp. 860–872, Sep. 2012.