Fall 12-18-2014

Computational Fluid Dynamics Analysis of an Ideal Anguilliform Swimming Motion

Charles Rogers
University of New Orleans, ctrogers@uno.edu

Computational Fluid Dynamics Analysis of an Ideal Anguilliform Swimming Motion

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Engineering
Naval Architecture and Marine Engineering

by

Charles Rogers
B.S. University of New Orleans, 2012

December 2014

Acknowledgment

This work for this thesis was made possible by Office of Naval research grant #N000141110830, and I am thankful for the opportunity to participate. I must also express my deepest gratitude to Dr. Brandon Taravella, without whom I would not have completed this work. Special thanks as well to Kelsey Murray for putting up with me during this process.

Contents

List of Tables

List of Figures

Abstract

There is an ongoing interest in analyzing the flow characteristics of swimming fish. Biology has resulted in some very efficient motions and formulating these motions is of interest to engineers. One such motion theory was written previously by Vorus and Taravella, and the University of New Orleans was given funds by the Office and Naval Research to test this theory. Computational fluid dynamics analysis was done using ANSYS Fluent to compare expected flow properties with the computed values.

Biomimetic CFD, anguilliform motion, ANSYS Fluent

1. Introduction

1.1 Review of Ideal Swimming Theory

The concept of a highly efficient motion is of great interest to hydrodynamicists. In general the work on this topic uses slender-body theory, which lends itself well to the modeling of ships and fish [11].

Several papers have been written on the topic of highly efficient motion by Lighthill. Lighthill's work is similar in its assumptions to Vorus and Taravella, using slender body theory and cylindrical cross sections [8]. One notable difference is that the Lighthill derivation in [8] includes a calculation for the transverse force (lift) on the body, whereas in Vorus and Taravella [16] this condition is explicit requested to be 0 net lift for all time and space. This 1960 Lighthill paper posits that the wave speed (the speed of the wave motion of the body) needs to be $^5/_4$ of the forward speed. Interestingly this is the same value that Vorus and Taravella [16] reached when calculating the advance ratio U, just defined inversely [16].

Lighthill in later work [9] delves further into high efficiency swimming motions, looking at carangiform motion as well as ideal anguilliform motion. This is a comprehensive paper on some of the basic mechanics of the motion types, including some of the logical reasoning behind the theory.

The origin of this work began with the paper by Vorus and Taravella [16]. The core idea outlined is that anguilliform swimming motion can be purely reactive, creating no net vorticity downstream. From a hydrodynamics standpoint, this is a highly desirable quality, essentially removing the induced drag from the equation [16]. What follows is a summary of the theory. The original paper includes a description of both a 2D and a 3D motion, but this research focuses solely on the 3D solution.

The genesis of the theory outlined by Vorus and Taravella [16] is a desire to capture the efficency of the anguilliform swimming motion. This motion is inherently different from thunniform motion, which relies on the movement of a tail fin instead of the undulating motion characteristic of eel. The main crux of the theory is that no circulation can be created by using a purely reactive thrust through the specific motion of the eel.

The theory stems from observations of anguilliform swimming motion, specifically juvenile lamprey. Several basic variables of the lamprey motion are described, such as the length, oscillation frequency, displacement wavelength, and mean tail displacement. Theses characteristics are important throughout the paper, as they help serve as conditions toward finding a solution.

One very useful value derived is the advance ratio U, defined in Vorus and Taravella [16] as the ratio of

the body forward speed U_0 and the displacement wave speed V such that $U = {}^{U_0}/_{V}$ [16].

From there the paper gives a conceptual analysis of this motion, using a 2D solution where the eel is modeled by an infinitely thin strip oscillating in the xy plane. The details of the 2D analysis [15] are not discussed here, but after the 2D analysis is done there is discussion of the 3D solution [16]. Slender-body theory is applied, solving all of the flow conditions for a 2D slice of the eel. This is shown in Figure 1.1, with the transverse displacement labeled h given as a function of time and longitudinal location. The perturbation potential as derived in [16] is given in 1.1.

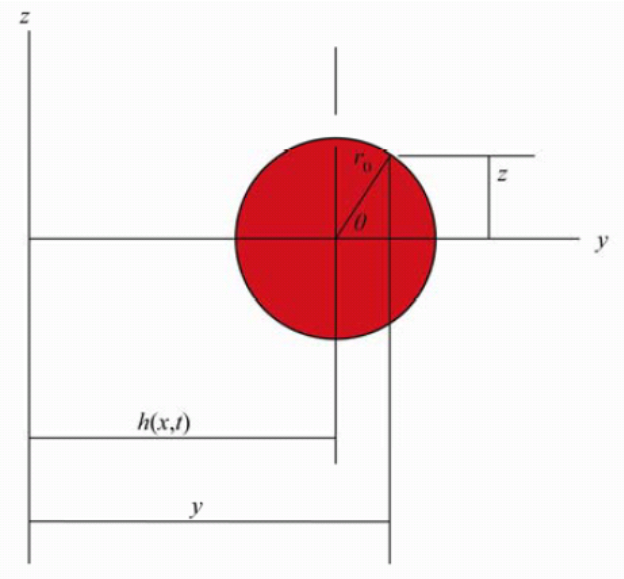

Figure 1.1: Cross section of the eel body used in calculations [16]

$$\phi(x, t, \theta) = -(h_t + U_0 h_x) r_0 \cos\theta \tag{1.1}$$

The above potential applies to the cylinder (in this case the eel) surface. Velocities in the x, y and z directions are derived from 1.1 and are given in 1.2, 1.3, and 1.4 respectively. Note that the x velocity has to include the forward speed U_0 in its formulation.

$$u = U_0 - (h_{tx} + U_0 h_{xx})\, r_0 \cos\theta \tag{1.2}$$

$$v = (h_t + U_0 h_x)\left(-\sin^2\theta + \cos^2\theta\right) \tag{1.3}$$

$$w = (h_t + U_0 h_x)\left(\sin\theta\cos\theta + \cos\theta\sin\theta\right) \tag{1.4}$$

Furthermore the linearized pressure on the surface of the eel can be derived from 1.1 and the Bernoulli equation to give 1.5.

$$p(x,t,\theta) = \rho k(x,t) r_0 \cos\theta \tag{1.5}$$

where:

$$k(x,t) \equiv h_{tt} + 2U_0 h_{xt} + U_0^2 h_{xx} \tag{1.6}$$

here subscripts denote derivatives. By extension two subscripts means two derivatives, and so forth. It is worth mentioning that the motion equation h is what is being found in Vorus and Taravella. While it appears in equations 1.1 through 1.4 it is not yet derived.

After several more steps the final motion is defined in equation 1.7.

$$h(x,t) = \Re[H(x)e^{-2\pi i t}] \tag{1.7}$$

with:

$$H(x) = i\Gamma\left(e^{2\pi i x} - e^{\frac{2\pi i x}{U}}\right) \tag{1.8}$$

And Γ defined as:

$$\Gamma \equiv \sqrt{\frac{C_T}{4\pi^3 r_0^2 (1-U)(1-\cos\frac{2\pi}{U})}} \tag{1.9}$$

In the above equations x is the longitudinal distance along the eel non-dimensionalized on eel length L, time t is non-dimensionalized on the ratio of displacement wave speed over length $^V/_L$, and r_0 is the radius of the eel non-dimensionalized on the length as well.

Also note that the advance ratio U appears in equation (1.9) [16]. A new variable is added in (1.9): the thrust coefficient C_T. Although there is no induced drag from the motion, viscous drag is unavoidable, so this must be taken into account. This thrust coefficient is defined as:

$$C_T = 2\pi C_{dv} U^2 r_0 \tag{1.10}$$

where C_{dv}, the viscous drag coefficient, is found using the Blasius drag prediction for a flat plate:

$$C_{dv} = \frac{D_v}{\frac{1}{2}\rho U^2 A} = \frac{1.33}{Re} \tag{1.11}$$

In equation (1.11) Re is the Reynolds number of the flow using the length of the eel as the characteristic dimension.

Two other derivations that is worth mentioning is the thrust distribution over time, which can be found by integrating (1.12) over the length. Note that since the position x is non-dimensionalized on length, this means integrating from 0 to 1.

$$C_T(x,t) = C_T(x) - \pi r_0^2 \Re[K(x)H_x(x)e^{-4\pi i t}] \tag{1.12}$$

The function $K(x)$ is defined as:

$$K(x) = -4\pi^2 H(x) - 4\pi^2 U i H_x(x) + U^2 H_{xx}(x) \tag{1.13}$$

Again, the subscripts in equation (1.13) define derivatives.

Looking back and plugging equations (1.8) and (1.9) into equation (1.7) and re-dimensionalizing the variables leads to a simple calculation for the displacement along the length of the eel:

$$h(x,t) = \Gamma \left[\sin\left(\frac{2\pi}{L} \left(\frac{V}{U_0} x - Vt \right) \right) - \sin\left(\frac{2\pi}{L} \left(x - Vt \right) \right) \right] \tag{1.14}$$

With the main equations in tow it is worth looking at the assumptions and simplifications that went into the derivations. The primary assumption is that the hydrodynamics were considered to be in ideal flow, namely an incompressible, inviscid, and irrotational fluid. The theory also goes on to assume that the unsteady boundary layer is laminar. It should be noted that the 3D solution mearly implies that the eel model has cylindrical cross sections and a finite radius; the motion itself (as evidenced by equation (1.14)) is still planar. It should also be noted that the theory requires the depth to remain constant over the length of the eel. This ties in to the fact that the theory is only for steady state eel motion [16].

Continuing the steady state solution requirement, the thrust mush equal the drag at all times to satisfy Newton's 3$^{\text{rd}}$ law (otherwise there would be acceleration in some direction and the solution would be transient). Observation of juvenile lamprey leads the paper to the conclusion that the animal will work toward a "minimum disturbance" to waste as little energy as possible, which works well with the assumption that the only drag in the model is purely viscous. By this assumption, C_D can be found as purely viscous drag as in equation (1.11) [16].

Three other requirements are stated for a non-trivial solution for 3D wakeless propulsion. These are

that there is no total transverse forces or "lift" for all times, the Froude efficiency is 1.0 for all longitudinal locations and times, and the maximum displacement of the motion is a minimum. The Froude efficiency is defined in the the theory to by output power over input power, and sectionioal Froude efficieny is given in equation (1.15). The third requirement seems a bit confusing at first, but makes sense with the heuristic approach mentioned previously. Essentially, the animal will undergo a motion that results in the minimum amount of work to conserve energy. As such, the maximum amplitude of the displacement is the minimum that the animal can do to maintain the motion [16].

$$\eta_i = \frac{-Uh_x(x,t)}{h_t(x,t)} \tag{1.15}$$

Further derivations and calculations of the theoretical motion can be found in the original paper by Vorus and Taravella [16].

1.2 Biomimetic CFD

With the advent and growth of computational fluid dynamics it makes sense that more and more hydrodynamicists have attempted to model the flow of various fish. In general, like this research, the modeling has stemmed from previous theoretical or experimental observations that is being tested in the computational environment.

As Vorus defines in [16], there is a difference between thunniform motion that involves circulation created from a fish's tail and anguilliform motion that is theoretically devoid of net circulation. Guan et al. [6] studied carangiform motion, a motion defined by rapid tail motions . This motion necessitates vortex shedding around the tail, different from what is outlined in Vorus. There is no great discussion of the theory behind this motion, simply a given sinusoidal equation.

Fish are not the only aquatic animal with an pertinent swimming motion. Liu et al. studied the swimming of tadpoles though CFD. The research was primarily looking at how the tadpole geometry allows it to have hindlimbs without a severe penalty to hydrodynamic efficiency when compared to typical fish shapes [10]. This motion is a sort of hybrid, with a long slender tail behind the large forebody of the tadpole.

Other research has been done on the motion of fish geometries behind bluff bodies. Work by Hannon [7] involved CFD analysis of the Karman gait, thunniform motion behind a bluff body optimized with vortices coming off of the body. The Hannon work was specifically done to recreate previous experiments computationally using ANSYS Fluent, however analyzing only a 2D solution. Similarly work by Sin et al. looked at motion behind a bluff body, in this case an eel shape behind the body that will move with the

Karman vortex sheet [12]. This model is also 2D and was more than just fluid flow analysis; it also involved the periodic undulation of the eel-like bar as a finite element model.

Some research uses a different solver to attack the problem. Cheng and Chahine used a boundary element method to calculate hydrodynamics around a swimming fish. [4]. This is a 3D solution that looked at the fish body as well as a wake sheet downstream.

Very similar to paper of Vorus and Taravella [16] was the work of Lighthill discussed in section 1.1, and an extension of this theory was tested by Boyer, Porez, Leroyer, and Visoneau [3]. This research involved the creation of a robotic eel as well as a CFD analysis. The geometry of this model is much closer to "eel" shaped compared to the previous papers discussed although it is not the idealized, circular cross sections outlined by Vorus, instead using elliptical cross sections that vary in size over the length.

This paper is perhaps the best comparison to the work involved here. The authors tested four different swimming motions: steady state forward (similar to the work of this thesis), turning, stopping, and a three dimensional "spiral". The setup especially was useful to see how other authors approached the problem [3]. The basic definition of the motion is also slightly different from that outlined by Vorus and Taravella. It can be seen from some of the figures in the paper that the head has some lateral motion, which is not present in the Vorus formulation.

Serving as sort of a forerunner to this research was previous work carried out at the University of New Orleans with the Vorus and Taravella paper. This involved the CFD analysis of a 2D solution very similar to the 3D solution used here [13].

It is important to note that the analysis of these papers did not look at just the flow theory and whether or not it was relatable to the work of Vorus and Taravella. A good deal of time was spent looking at the methodology used in the setup of the CFD simulations, where it was readily available. Characteristics such as mesh setup and boundary conditions were analyzed to see which would be most applicable to the Vorus model, if there were any pitfalls regarding setup of the problem, etc.

The first and primary takeaway from these CFD papers is the mesh setup. Looking at what has been done previously can aid in the creation of the mesh for this research. Some papers chose a structured mesh around the eel, then unstructured in the far field. Others chose either a completely structured or unstructured mesh. Several papers ([3], [10], and [12]) use a sctructured mesh around the body that is easy to create because the body starts off in a straight line, i.e. the centerline of the body is parallel to the x-axis. Note from equation 1.14 above that the theory outlined by Vorus and Taravella does not start with the eel straight; rather the eel is already displaced at $t = 0$. This would seem to restrict the possible creation of the mesh for the simulations. A similar example to the setup that was used here is the example of Guan, where the mesh immediately around the body was unstructured and the mesh was structured in the far field [6]. The

boundary element method outlined by Cheng is an obvious outlier, as the domain used in that paper is the surface of the fish body and a wake sheet on centerline rather than a fully meshed domain. More on this is discussed in section 3.2.

It also intersting to note how the 3D papers defined the overall domain. Guan et al meshed a full domain around the body (in front, behind, to the left and right, above and below). On the other end of the spectrum Boyer et al took advantage of symmetry to model just half of the eel and domain (just above, not below). This was helpful information when constructing the domain for the analysis, as fewer cells help to speed up the calculations.

When possible it is also good to see how these papers dealt with the problem of the moving body boundary. The three papers that used ANSYS Fluent ([6], [7], and [13]), the same program that is used in this research, all used the `DEFINE_GRID_MOTION` macro to move the body boundary. Most of the research into this biomimetic theory also involved a fully described motion. That is, instead of using the fluid flow as an input to the CFD calculation the motion is fully described and the resulting fluid domain is the output (the ideal fluid solution is usually the input to solve for the motion in the first place on paper). This is discussed more in section 3.3.

2. Research Overview

2.1 Scope of the research

This research aims to simulate through CFD the theory outlined by Vorus and Taravella [16]. Data from the simulations is to be used to either validate or invalidate the theory.

The main points examined from the theoretical motion are the assumptions, that is, which assumptions are valid and which are invalid. Theoretical thrust, pressure distribution along the eel, and the presence of vorticity are compared to the simulation results to see if there is agreement.

The simulation procedure is to test inviscid solutions at the design speed of the eel. An off-speed test is also performed under inviscid conditions to see if there is vortex formation. The basic procedure involves five main steps:

1. Create the model geometry
2. Build an appropriate mesh for the geometry
3. Create a UDF file the will properly move the mesh to match the theoretical motion
4. Setup a valid simulation
5. Post-process and analyze the results

The first step, to create the model geometry, is started in Rhino. Model completion and all following steps take place in the ANSYS environment. Workbench for the model completion and mesh creation, Fluent for the calculations, and Fluent/CFD-Post for analysis and post-processing.

3. Flow Solver and Solution Methodology

3.1 Solver Theory

ANSYS Fluent CFD software was used to carry out the calculations. Fluent uses a finite volume method to discretize and solve the Navier-Stokes equations, mass conservation, and if required the energy equation.

From the theory used in Fluent, the generalized, conservative differential form of the continuity equation is:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \overrightarrow{v}) = S_m \tag{3.1}$$

In the above, per [2], the first term on the left is the change, the second is the convective term resolving mass flow across boundaries, and the S_m term is a source term that is under the control of the user.

Similarly the general conservation of momentum equation used in Fluent is:

$$\frac{\partial}{\partial t}(\rho \overrightarrow{v} + \nabla \cdot (\rho \overrightarrow{v}\,\overrightarrow{v}) = -\nabla p + \nabla \cdot (\overline{\overline{\tau}}) + \rho \overrightarrow{g} + \overrightarrow{F} \tag{3.2}$$

In the above equation from [2] p is the static pressure, $\rho \overrightarrow{g}$ is the gravitational body force, and $\overrightarrow{F}$ is the collection of external body forces. The stress tensor $\overline{\overline{\tau}}$ is defined in 3.3 as:

$$\overline{\overline{\tau}} = \mu \left[\left(\nabla \overrightarrow{v} + \nabla \overrightarrow{v}^T \right) - \frac{2}{3} \nabla \cdot \overrightarrow{v} l \right] \tag{3.3}$$

Equations 3.1 and 3.2 can be rearranged (discussed at length in [2] and [14]) as a transport equation, written generally [14] as:

$$\frac{\partial (\rho \phi)}{\partial t} + \nabla \cdot (\rho \phi \overrightarrow{v}) = \nabla \cdot (\Gamma \nabla \phi) + S_\phi \tag{3.4}$$

Looking at 3.4 the first term is the rate of increase of the property ϕ within the differential element, the second term is the net outflow rate of ϕ from the element, the third term is the increase rate in ϕ due to diffusion Γ, and the fourth term is the rate of increase due to sources. Equation 3.4 can be integrated for an arbitrary control volume and discretized spatially to get to equation 3.5 [2]. The spatial discretization is discussed more in section 3.5.

$$\frac{\partial(\rho\phi)}{\partial t}V + \sum_{f}^{N_{faces}} \rho_f \overrightarrow{v}_f \phi_f \cdot \overrightarrow{A}_f = \sum_{f}^{N_{faces}} \Gamma_f \nabla\phi_f \cdot \overrightarrow{A}_f + S_\phi V \tag{3.5}$$

In 3.5 it is evident that the convective and diffusive terms are summations of the flux through all boundaries of the volume, while the rate of change term is depended on time (which is not discretized in the above equation but is a simple calculation). It is important to note now that the transport property ϕ in equation 3.5 is now found on a face. In this way it must be resolved not only for a single volume but for all volumes that are bounded by that face. Extrapolating this out to many thousands of small cells, the result is a large set of equations that must all be solved simultaneously for the correct transport property. Traditional equation 3.5 needs to be linearized to solve the equations, resulting in a large set of linear equations [2].

There are several different methodologies in Fluent to solve the discretized equations, each with its own pros and cons. These four algorithms available for the pressure-based solver are SIMPLE, SIMPLEC, PISO, and Fractional Step.

SIMPLE stands for Semi-Implicit Method for Pressure-Linked Equations. The algorithm involves four main steps. The initial solution is a guess at what the correct values for the pressure, velocity, or other transport property are in the domain. The first step is to solve the discretized momentum equations. Second, a pressure correction equation is solved to get the difference between the correct pressure and the initial guess. Thirdly, The pressure is corrected and this information is used to correct the velocities as well. Fourth, all other transport equations are solved. Then the solution is check for convergence. If the solution has converged, the calculations are complete. If not, the corrected pressure, velocity, or other transport values become the new inputs and the entire process is iterated again [14]. The SIMPLE algorithm can be thought of as sort of the "basic" algorithm.

Along with SIMPLE Fluent has SIMPLEC, SIMPLE-Consistent. The difference between the two algorithms is in the correction equation on the face flux. In certain cases where the pressure and velocity are coupled is an issue the SIMPLEC algorithm can converge faster than just the SIMPLE algorithm [2].

PISO stands for Pressure Implicit with Splitting of Operators. This method is essentially a continuation of SIMPLE with additional corrector steps to the calculations. There is extra computational effort as there are now more equations to solve with each iteration, but the PISO scheme can reach convergence with fewer iterations than the SIMPLE or SIMPLEC scheme, improving overall computation time [2]. Figure 3.1 shows a breakdown of steps for the PISO scheme.

The Fractional Step scheme is used with non-iterative time-advancement. This method is distinctly different from the previous SIMPLE algorithms in that the test for convergence does not take place at the end of the time step but instead there are smaller inner iterations at various stages of the calculation [2].

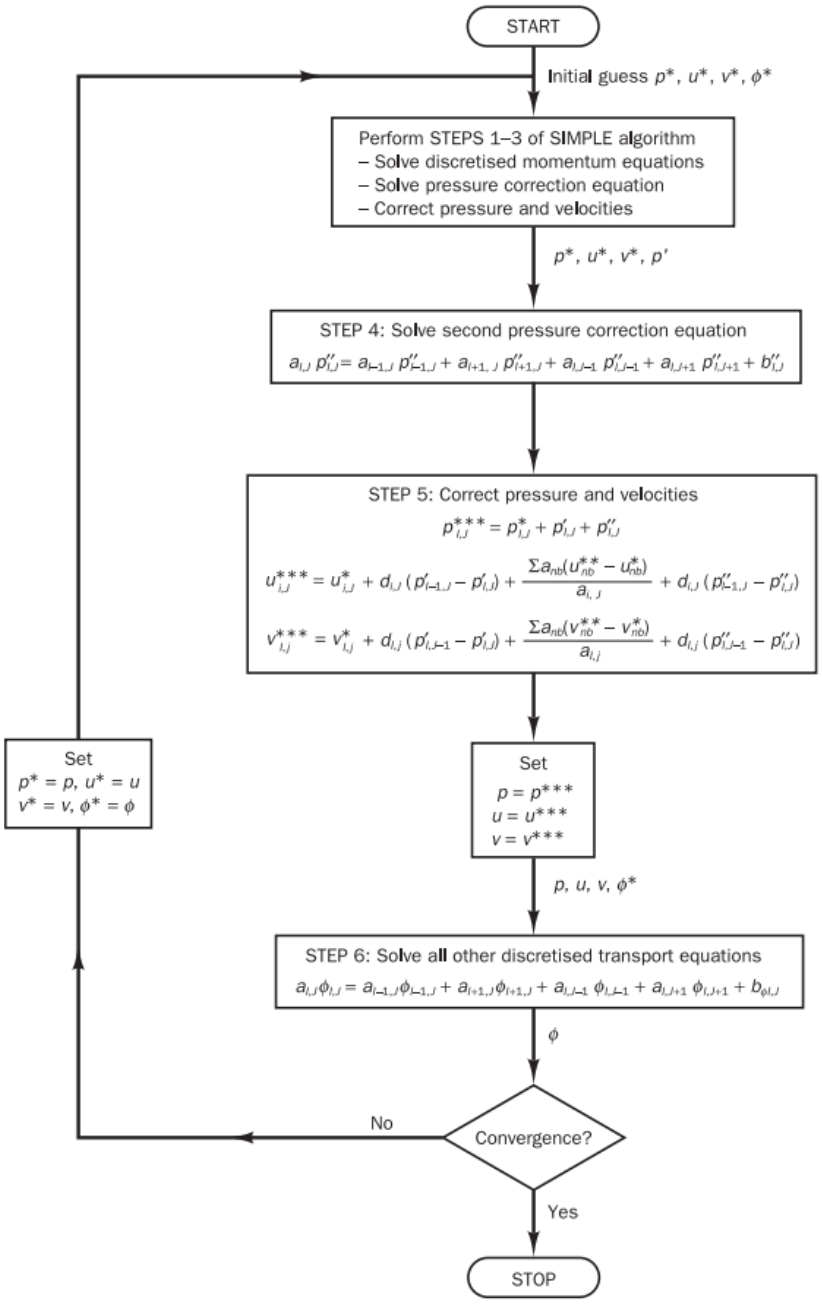

Figure 3.1: Breakdown of steps in the PISO algorithm [14]

Which solution algorithm to use in this eel application involves some decision making, specifically: which algorithm is best suited to solve the eel simulation? Consulting with information from [1], [2], and [14], it is apparent that there is no clear cut "best" algorithm for certain applications. Some perform better at certain things, but for the most part the solution method needs to be chosen on a case-by-case basis. Looking at the methods available and iterative PISO algorithm was chosen for its corrections to help solve with the

potential distortion around the eel.

3.2 Mesh Creation

The creation of the mesh is an integral part to the solution. The mesh is the domain of the solution, a discretized space that defines the volume for the calculations. It is the spatial discetization of the governing equations, and as such it can affect the outcome of the solution. The desired domain is divided into many individual cells of varying shapes, with nodes at the corners. In 3D these cells all have to be closed volumes, such as hexahedrons or tetrahedrons.

There are two general categories that meshes can be classified into: structured and unstructured. Structured meshes, as the name implies, are made up of a structured layout. Cells in a structured mesh are hexahedral, creating a grid both length-wise and width-wise in the domain. Unstructured meshes, on the other hand, are not laid out in a grid structure. Instead, triangular geometries are used alongside the rectangular shapes to discretize the domain. This allows for the creation of more complex geometries that cannot be simply modeled with a purely rectangular faceted mesh.

The different methods of meshing have various other pros and cons that need to be considered with regard to the geometry, computation time, and numerical diffusion. Simple geometries can be created with a structured mesh, but this can oversimplify more complex designs. As such an unstructured mesh may be required to adequately represent the desired geometry. The time it takes for the computation to reach convergence can be greatly affected by the layout of the mesh. More cells means more calculations, increasing the time required for the simulation [1].

The key consideration that must be made when constructing the mesh however is numeric (or false) diffusion. Numeric diffusion is not physically real, but a side effect of discretizing the equations. It is essentially the compounding of truncation errors in the calculation, something that cannot be avoided and therefore needs to be properly considered when creating the mesh. These errors manifest as increasing diffusion and therefore are more prominent when the simulation is convection-dominated. Numeric diffusion is inversely related to the resolution of the mesh, i.e. a coarser mesh will have more false diffusion than a more refined mesh [1]. A good analogy for this is the concept of higher order terms in regards to Taylor series; the more terms (cells in the mesh) the more precise the solution (the less false diffusion is captured). Another way to combat false diffusion is to align the cells in the mesh with the flow. This is easily done with hexahedral meshes, but tetrahedral cells will never be aligned with the flow. This will also change if the flow has complex characteristics, such as swirling vortices [1].

It is apparent that the creation of the mesh involves trade-offs between complexity, computation time,

and built-in errors. The geometry needs to be modeled with sufficient detail so as to properly represent the swimming eel, but at the same time a minimum amount of nodes is desired to allow for quicker computation time, but there needs to be enough resolution to minimize numeric diffusion, and so on. The point of this catch-22 is that engineering judgment needs to be used the design of the mesh. It was decided that the region around the eel should be meshed with tetrahedral cells to properly define the geometry, and farther away from the eel there would be hexahedral cells. Essentially the domain is divided into two "boxes". The eel itself is inside a box of tetrahedral cells, and outside of this box out to the edge of the simulation domain there are hexahedral cells in the mesh. Figures 3.2 through 3.5 show several views of the mesh.

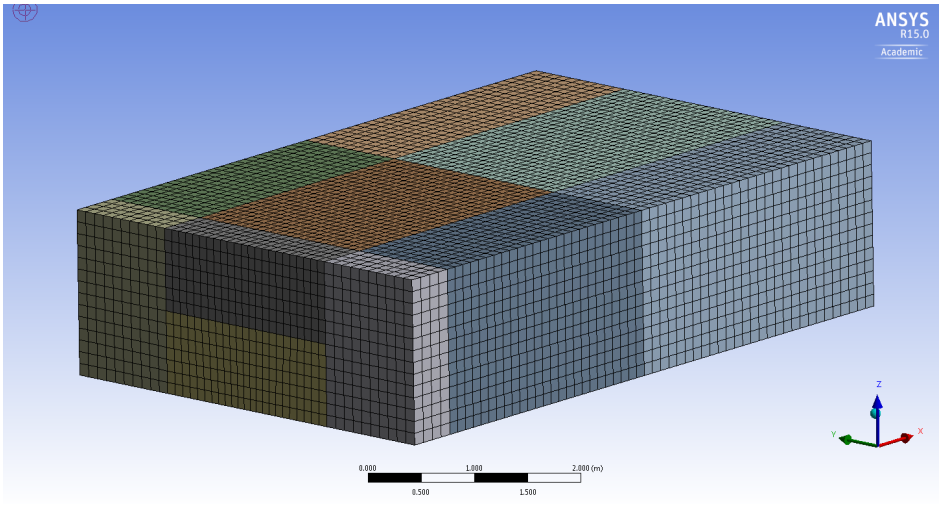

Figure 3.2: Example of the mesh used in the calculations

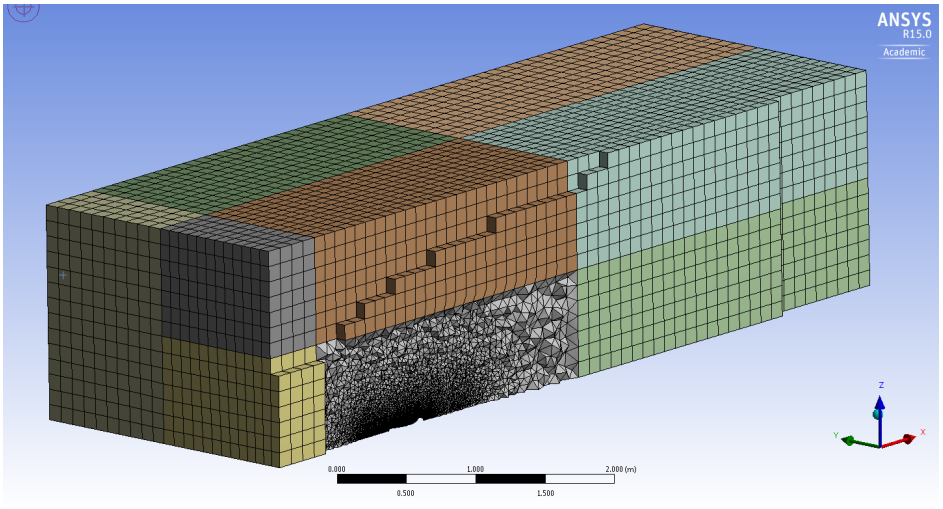

Figure 3.3: View down centerline of the mesh

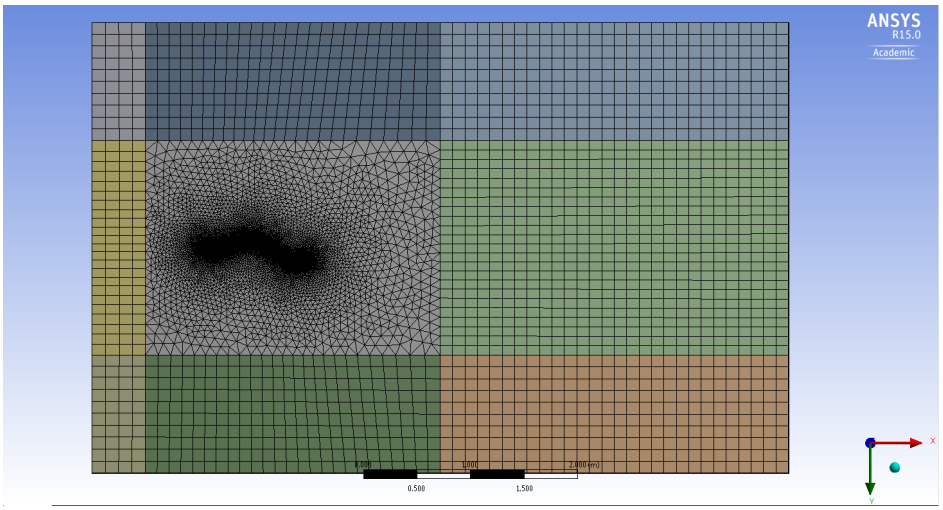

Figure 3.4: Example of the mesh used in the calculations with 162884 modes

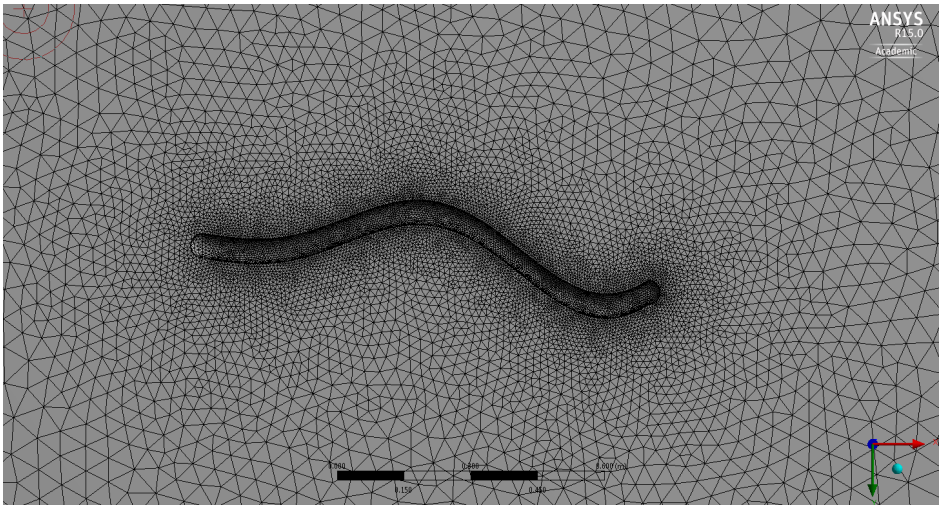

Figure 3.5: Zoomed view of the mesh around the eel model

3.3 UDF Creation

The required motion of the eel has to be told to the solver in order for the proper movement to take place. This is done in Fluent by using a User Defined Function (UDF). UDFs are used to enhance certain features of Fluent, such as custom boundary conditions or post-processing. In this case the UDF is constructed to apply the mesh movement to match the theoretical motion. UDFs are written in the C programming language, with the addition of a UDF header file for the Fluent macros.

There are four main `DEFINE` macros in the `udf.h` header file used by Fluent. These are `DEFINE_CG_MOTION`, `DEFINE_GEOM`, `DEFINE_GRID_MOTION`, and `DEFINE_SDOF_PROPERTIES`. Each one has a proper application and use, so choosing the proper macro is a requirement for getting the correct eel motion to the solver.

The DEFINE_CG_MOTION macro works by the UDF sending Fluent the linear and angular velocities of a body, and Fluent then uses rigid body motion to update the mesh position on the dynamic zone at the next time step. This is unnecessary for the required motion, since equation 1.14 is continuous. This could be useful for modeling the motion of a segmented eel of rigid structure, but not needed for these requirements [5]. It would require a dedicated analysis of the segement properties of the eel and lies outside the scope of this project.

The DEFINE_SDOF_PROPERTIES macro is used to define different properties of a body for the six-degree of freedom solver in Fluent. These are typical body properties: mass, moment of inertia, product of inertia, and some information about moment properties and external forces. This macro is best used for simulating how a body interacts with a flow field. The six-degree of freedom solver in Fluent uses the flow around the body and the body properties to move the dynamic zone to the next time step. Essentially the flow defines the motion that the dynamic zone will take [5]. This is the opposite of the eel theory of Vorus and Taravella, where the eel motion accompanies a specific flow field.

The DEFINE_GEOM macro is slightly different from the previous two macros discussed. Instead of operating on a dynamic zone (the zone that is the genesis of the movement of the mesh) this macro applies to deforming zones. That is, this macro works on the zones that need to move around the dynamic body. Nodes in the deforming zone are given a specific position after undergoing re-meshing, taking a new position outlined from the user in the UDF [5]. This would get very unwieldy in the eel mesh, where the main deforming zone is a large, unstructured mesh around the eel. There would be many many nodes needing the re-positioning, and no good theory to outline the proper placement of the nodes (and it would change depending on the mesh refinement). Therefore it was not used for this simulation.

The DEFINE_GRID_MOTION macro works by taking input from the UDF to move the nodes on the dynamic zone to an updated position for the new time step. All the node positions are updated independently of one another on the dynamic zone, instead acting as a function of the data in the UDF [5]. This method allows for specific movement of dynamic mesh zones, with the caveat that caution must be taken to ensure that the body retains the proper shape (since every node is independently updated by user input).

The method used to update the eel motion for the simulations was therefore done with the DEFINE_GRID_MOTION macro. It was the best way to update the nodes given the characteristics of the theoretical motion equation. Moreover, the three other macros of dynamic mesh movement and not applicable to the problem. The full UDF file used for the simulations can be found in appendix 6.1.

There is a unique feature of the theoretical motion outlined by Vorus that makes the creation of the UDF an integral part of the solution. Recall equation (1.14), the final motion equation. While this equation involves considerations for the thickness of the eel (hence 3D) it only specifies the motion of the eel centerline.

Looking again at Figure 3.5, it is clear that the centerline is not part of the model. Instead it is the eel surface that is required to move. For the theoretical motion this is fine, as the circular cross sections are supposed to remain in one plane. However this is not the case when trying to model the eel more realistically. This kind of motion requires new equations that properly define the motion of the eel surface such that is stays perpendicular relative to the centerline at all points. That is, the eel should have a constant perpendicular cross section along its entire length. This methodology was not implemented in the simulations for the project, however, and instead all nodes on the eel body were moved with the centerline motion outlined in equation (1.14). The one exception is the head and tail hemispheres. The head remains stationary per the Vorus theory and the tail simply moves with last node in the length of the eel.

3.4 Dynamic Meshing

The setup of the dynamic mesh is an integral part of the solution. This component is dependent on the makeup of the mesh described in section 3.2 and can change depending on the mesh. Like most other settings available in Fluent, there are several options that need to be considered to get the best possible results in the simulation. Fluent has three main options to update interior volume meshes when there is a dynamic boundary: smoothing methods, dynamic layering, and remeshing methods. Some are only applicable in certain applications and several can be switched "on" at once, so it is imperative to know which method are viable for the required situation [1]. Having the proper dynamic meshing methods can help avoid negative cell volume errors, which derail a run immediately.

Dynamic layering adds or removes layers of cells next to a moving boundary based on the height of the cell layer adjacent to the boundary. This method however is only usable with either wedge or hexahedra cells adjacent to the face [1]. Recall from section 3.2 that the mesh around the eel is purely tetrahedral; therefore the dynamic layering methods are not applicable and not used for this simulation.

Smoothing methods encompass spring-based, diffusion-based, Laplacian, and methods. The spring-based smoothing treats the edges betweeen two nodes as springs. The movement of the boundary nodes create a "force" that, using hook's Law, is used to calculate a displacement for all the interior nodes in the deforming boundary. Spring smoothing is applicable to all deforming zones with dynamic boundaries and best used with tetrahedral cells, but can be used for non-tetrahedral cells. Diffusion-based smoothing uses the diffusion equation and mesh velocity to move the interior cells. Boundary conditions on the diffusion equation are such that the cell motion is tangent to the dynamic zone and from there boundary motion is diffused into the interior of the mesh. Diffusion-based smoothing is an alternate to the spring method, applicable to all cell types. Diffusion-based smoothing is more expensive computationally but makes up for this by generally

resulting in a better quality mesh and allowing for greater zone motion before breaking down [1]. These two methods were deemed most applicable to the eel application.

The other smoothing methods, Laplacian and boundary layer smoothing, were not used for this simulation. Laplacian smoothing is the simplest smoothing method in Fluent, but it can lead to poor quality in the cells. Boundary layer smoothing is used to move the boundary layer of cells with the deforming zone [1]. Since the the boundary layer meshing technique was not used with the eel mesh, this method was not applicable.

Remeshing methods are used when the dynamic zone displacement is large compared to the cell size at the boundary. In these cases, when the mesh moves, the cell quality around the boundary can deteriorate quickly, creating negative cell volumes. To avoid these issues Fluent identifies problem cells that violate a desired set of skewness or size criteria and locally remeshes the problem cells, interpolating the solution in the new cells from the old cells. If the new cells also fail the skewness or size criteria they are discarded instead of integrated into the mesh [1].

Fluent has five main methods for remeshing: local cell, local face, face region, CutCell zone, and 2.5D [1]. Like the smoothing techniques, some of these are applicable only to certain cell shapes and can be discarded based on how the eel mesh was created. Local cell and local face remeshing were the only techniques used for the eel simulation.

Fluent uses the dynamic meshing settings in conjunction with each other to try and optimize the mesh at every time step. Fluent decides, based on the enabled models, which is best for the required deforming zone and then applies the smoothing techniques, then remeshes as necessary. For the eel simulations, both diffusion and spring-based smoothing were tested to see how the resulting mesh deformed over the time of the simulations. The same initial mesh and time step size was used for this comparison. Figures 3.6 and 3.7 show the resulting deformation for the diffusion method.

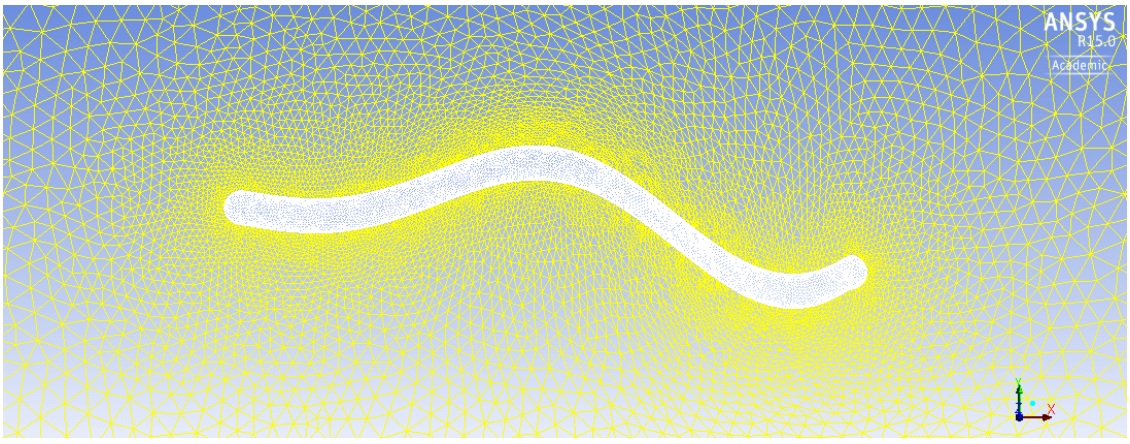

Figure 3.6: Deformed mesh using diffusion remeshing at $t = 0.5$

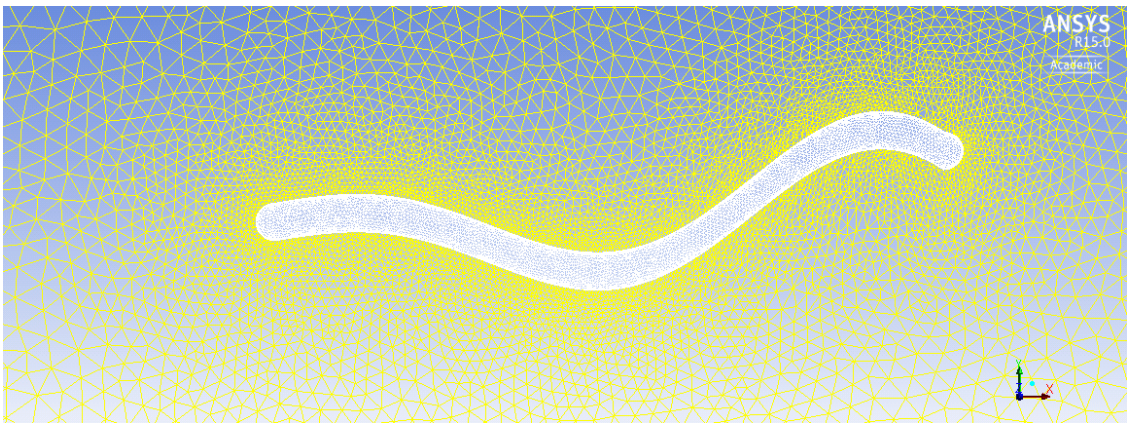

Figure 3.7: Deformed mesh using diffusion remeshing at $t = 1.0$

The spring smoothing resulted in a "cone" around the eel, but not too much deformation outside of the eel motion. The diffusion method seems to result in a "twisting" of the mesh extending out beyond the eel, as can be seen in Figure 3.6, but the overall symmetry plane looks better than with the smoothing method. Figure 3.6 shows the kind of twisting that happens as the eel moves, but note that the mesh returns to a very similar configuration to the starting setup after the cycle is complete in Figure 3.7. However, it is worth to check the overall number of nodes still left in the model after the simulation. These numbers are shown in Table 3.1.

Eel mesh characteristics over time

t	Nodes
0.0	198901
0.25	56604
0.5	56514
0.75	56153
1.0	55779

Table 3.1: Node numbers over time

It should be noted that the time values in the first column of Table 3.1 are non-dimensional over the cycle of the motion. It is clear that the remeshing settings lead to a significant drop in the number of cells around the eel over time. There also appears to be asymptotic behavior with the number of nodes as the rate decreases as well. Taking the information from Table 3.1 and looking at Figure 3.6 it is clear that that majority of nodes and cells that are being merged are in the vertical (z-direction normal to the symmetry plane) domain around the model.

3.5 Solution setup

The steps used to set up the solution involved the same kind of engineering judgment as the set up of the mesh, dynamic mesh, and solver settings. This includes the cell boundary conditions, the spatial discretization, the solution initialization, and other flow models.

The boundary conditions on the problem can have a large effect on the solution. In this case the goal was to properly model the theory, in which case the far field condition is that there is uniform flow. Fluent has many different options for boundary conditions with varying applicability, but the three used for the simulations were a wall condition, velocity inlet, and a symmetry condition.

The eel body itself is given a wall condition, used between solids and fluids. No flow perpendicular to the boundary is enforced, as well as a slip/no-slip condition depending on whether the flow model is viscid or inviscid. Recall now from section 3.2 that the mesh is rectangular, with half of the eel on one side. All boundaries, except this one that has the eel, are set to a velocity inlet. This is to model the perpendicular flow in the far field, away from the eel. The required velocity of the flow (i.e. the forward speed of the eel U) is input here in the longitudinal direction of the eel (along its length). At the same time the associated dynamic pressure from Bernoulli's equation is also required to be satisfied on the boundaries. The only boundary this is not done for is the "bottom", the boundary where the eel model is located. On this boundary a symmetry condition is desired. This is used to help shorten the computation time. The are some caveats that are associated with the symmetry boundary condition however. Namely, there is zero normal velocity and zero gradients normal to the symmetry plane. In some flow models this is a valid assumption, but in others this can be a hindrance. There is also zero shear stress along the symmetry plane, so there is slip along the boundary for viscous calculations [1].

The spatial discretization of the governing equations in Fluent is possible several different ways. There are also several different equations that need discretization including the momentum equations and continuity. The different methodologies differ in computational time and accuracy depending on the mesh flow characteristics.

The gradients of the variables in the transport equation 3.5 are found via a Green-Gauss node-based method. This method averages the values of the nodes on a face to get the face value, which is better suited to the unstructured mesh surrounding the eel (see Figure 3.5) [2]. The pressure interpolation is done via the standard procedure in Fluent, which was deemed applicable to this simulation. For momentum, a second-order upwinding scheme is desired, again because of the unstructured nature of the mesh around the eel. This method is more computationally intensive than the first-order method but gives more accurate results [1].

Since the Navier-Stokes equations are mixed parabolic-hyperbolic partial differential equations it is necessary to have initial conditions as well as the proper boundary conditions mentioned previously. The entire flow field was initialized to eel forward speed U as a condition of the theory, similar to the way the boundary conditions were set.

The time step size used for the calculations has major implications to the solution. There are two competing goals for this decision: remeshing and computation time. From a standpoint of remeshing it is desirable to have a smaller time step to make the remeshing calculations simpler and to avoid negative cell volume errors (when a cell normal flips directions and points the wrong way giving the cell a "negative" volume). However from a standpoint of computation time a large time step is desired. Larger steps lead to fewer calculations, which means a decrease in computation time. There is therefore a trade off when setting the time step size.

For the calculations two time step sizes were tested, 0.001 and 0.0005 seconds. Comparisons can be made between the two data sets to ensure that the time step size is not affecting the calculations.

4. Results and Discussion

4.1 Experimental Inputs

There are several required inputs to calculate the motion for the simulations. This includes setting the length of the eel, the forward speed, the advance ratio, the radius, and also an initial drag coefficient estimate. These input values are shown in Table 4.1.

Eel motion calculation data		
Length L	m	1.0
Forward speed U_0	$\frac{m}{s}$	0.5
Advance ratio U	$-$	0.8
Displacement wave speed V	$\frac{m}{s}$	0.625
Radius r_0	m	0.028
Thrust coefficient C_T	$-$	0.00021220
Γ	$-$	0.10445948

Table 4.1: Inputs for the motion calculation

Above variables were calculated using the formulas in section 1.1. These inputs are somewhat arbitrary; these values were chosen because they were used in examples in the original paper [16]. The main difference is the radius of the eel, which is different here from that which was used in Vorus and Taravella [16].

4.2 Fluent output

The main points of comparison between the theory and the CFD is between the velocities on the surface, the pressure, and the occurrence of vorticity downstream of the eel.

The velocity graphs can be seen in Figures 4.1 through 4.25. Velocities were checked at fractions of 0.25, 0.5, and 0.75 of the length at quarter, half, three-quarter, and full cycle. For the given inputs in Table 4.1 and using relationships outlined in section 1.1 this lead to times of 0.4, 0.8, 1.2, and 1.6 seconds, respectively. theoretical velocities were calculated according to equations 1.2 through 1.4.

These figures show that the velocity on the body compares well to the theoretical values. It is clear that the change in time step size did not affect the solution values, and they clearly show the trend outlined by the theoretical curve.

4.2.1 Velocity Plots

X velocity

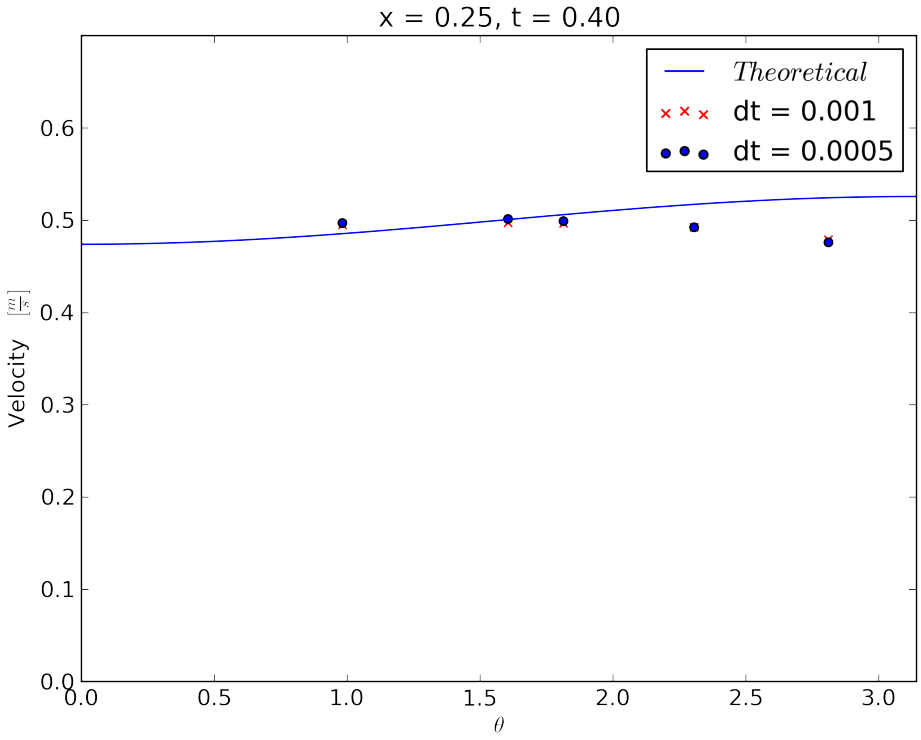

Figure 4.1: X velocity at $x = 0.25$, quarter cycle

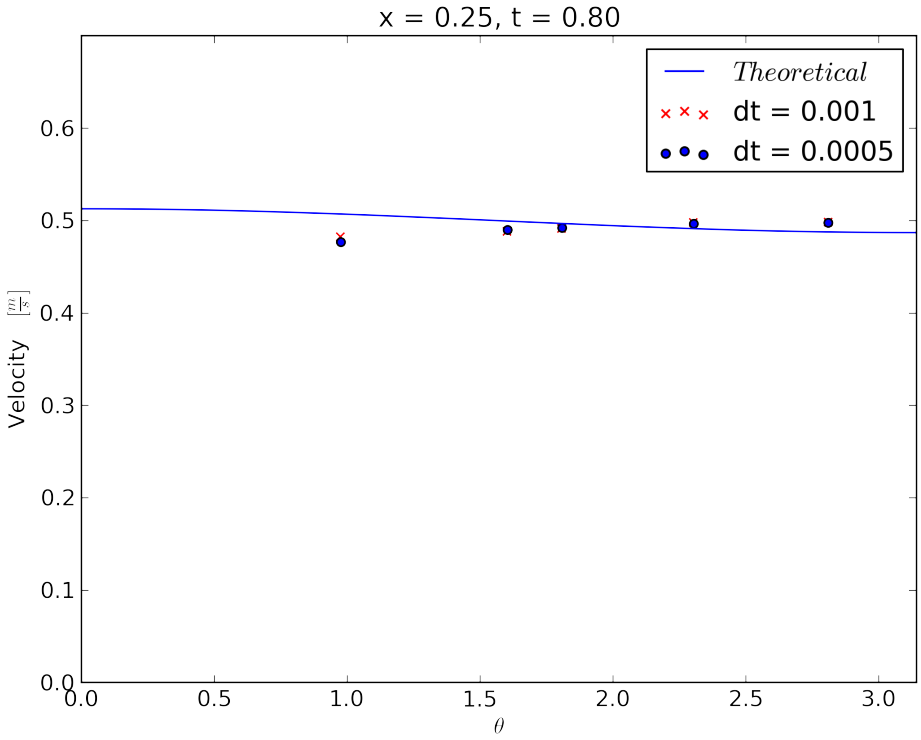

Figure 4.2: X velocity at $x = 0.25$, half cycle

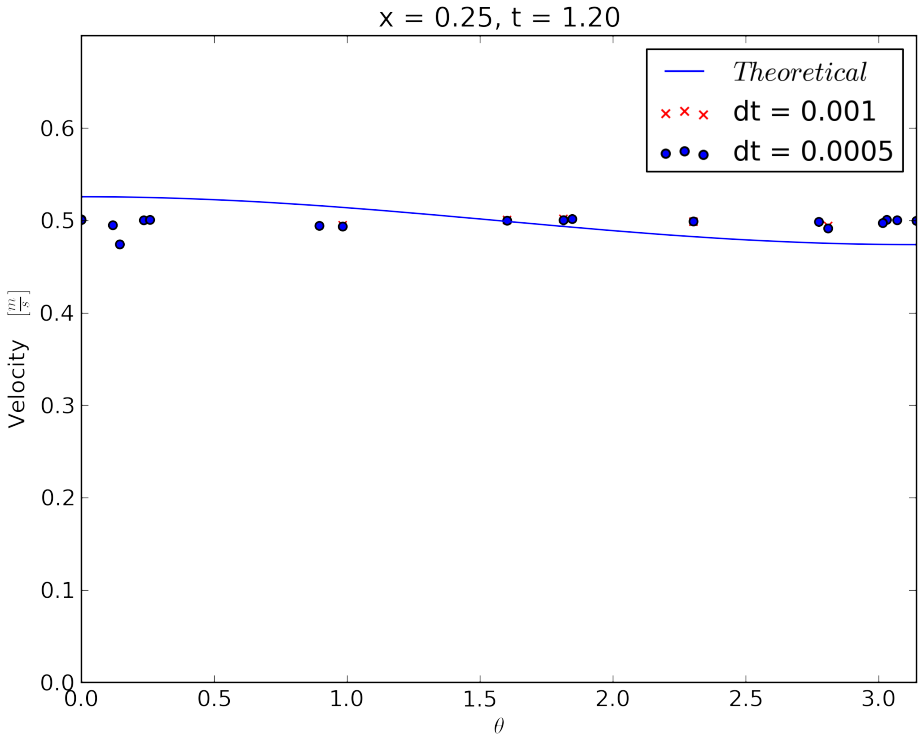

Figure 4.3: X velocity at $x = 0.25$, three-quarters cycle

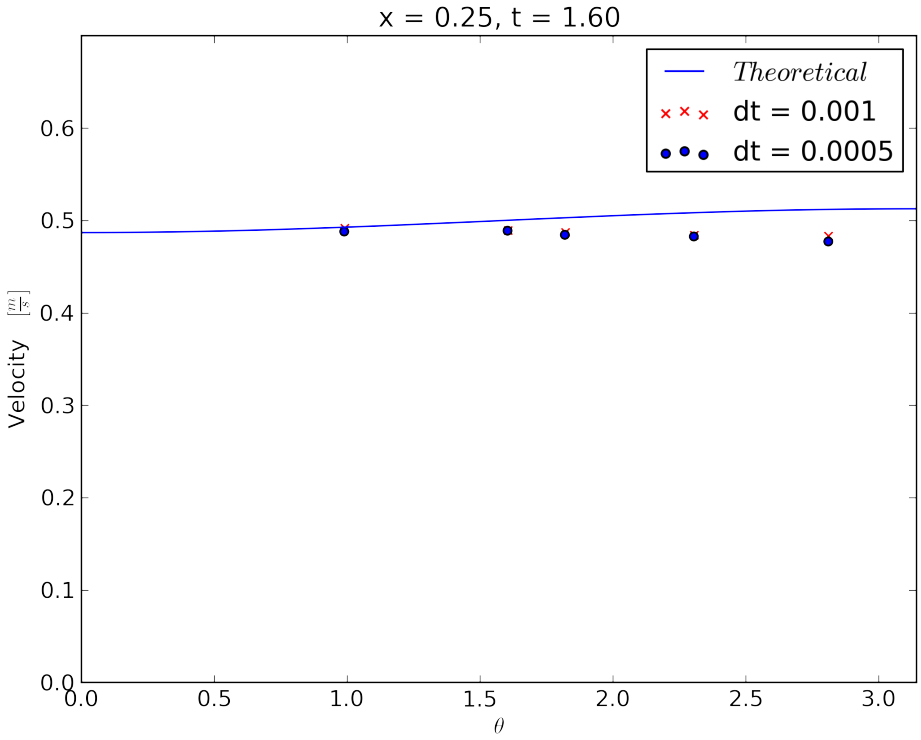

Figure 4.4: X velocity at $x = 0.25$, full cycle

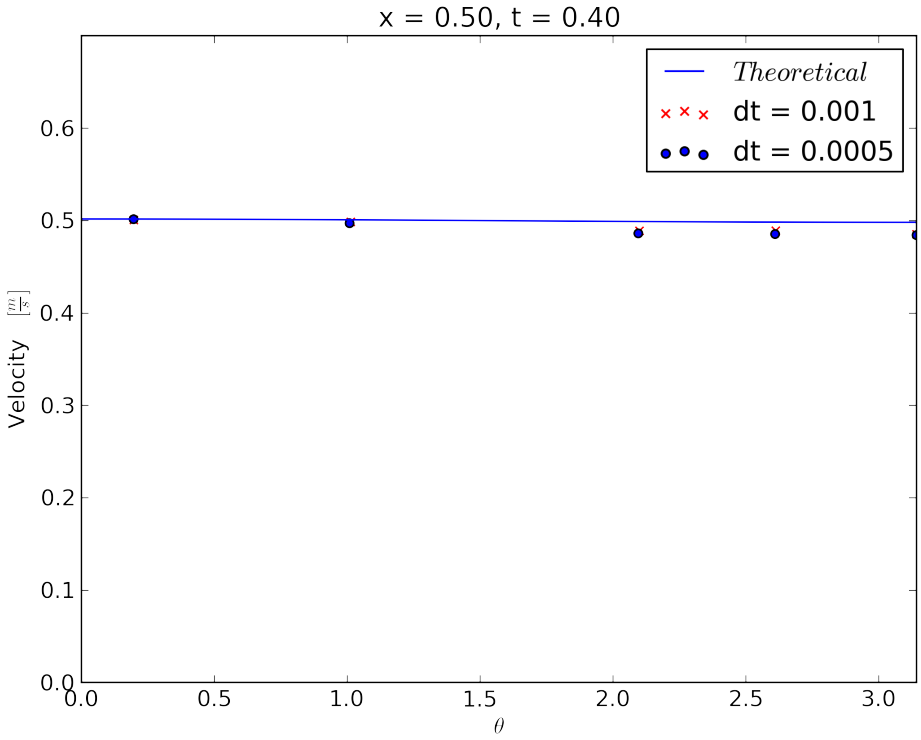

Figure 4.5: X velocity at $x = 0.50$, quarter cycle

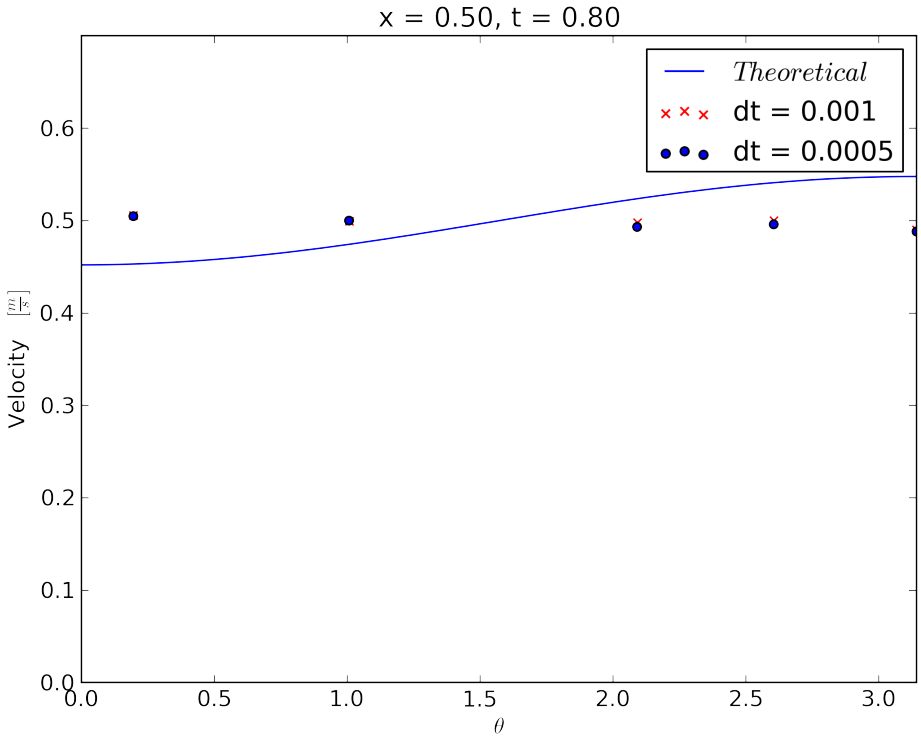

Figure 4.6: X velocity at $x = 0.50$, half cycle

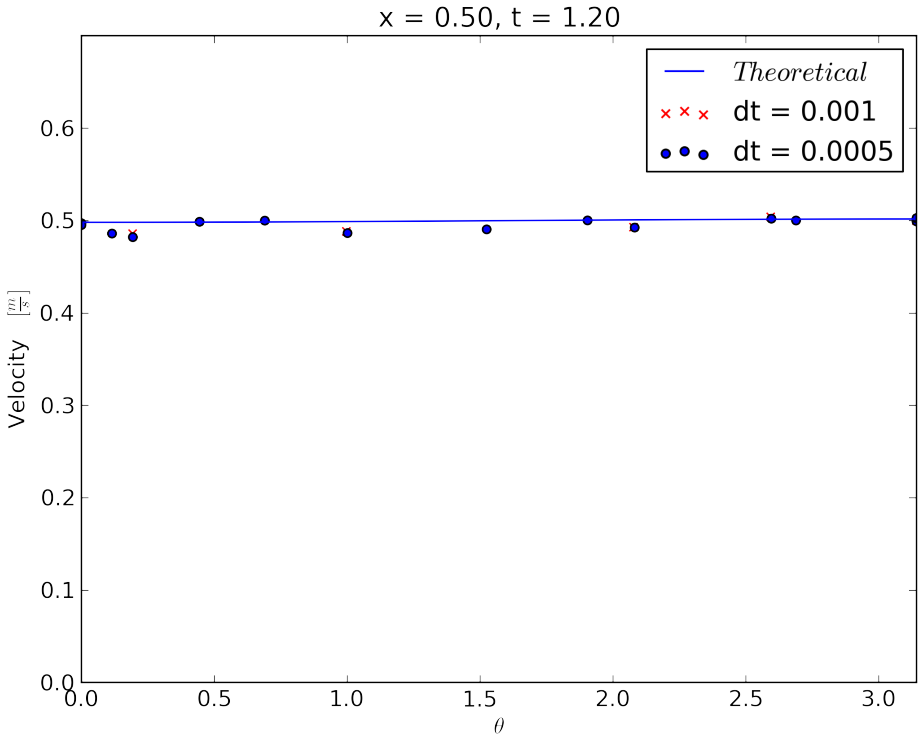

Figure 4.7: X velocity at $x = 0.50$, three-quarters cycle

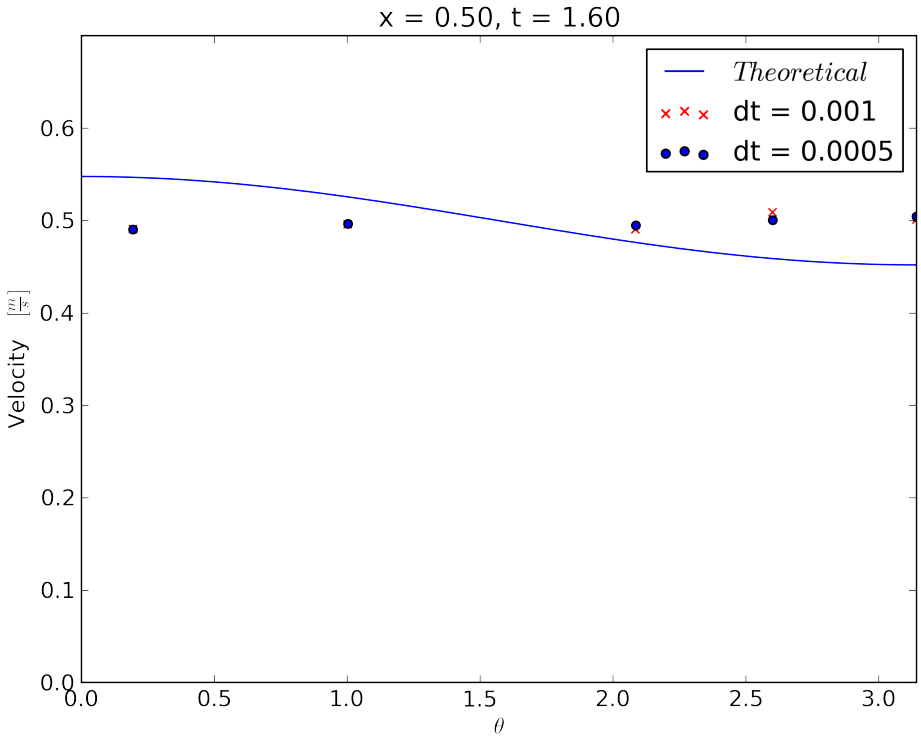

Figure 4.8: X velocity at $x = 0.50$, full cycle

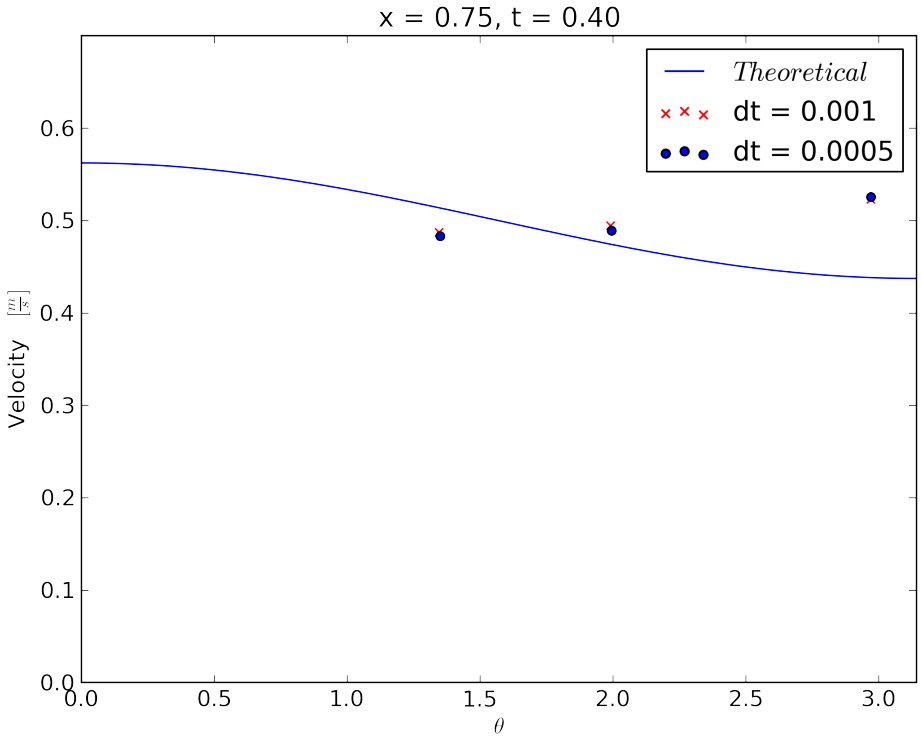

Figure 4.9: X velocity at $x = 0.75$, quarter cycle

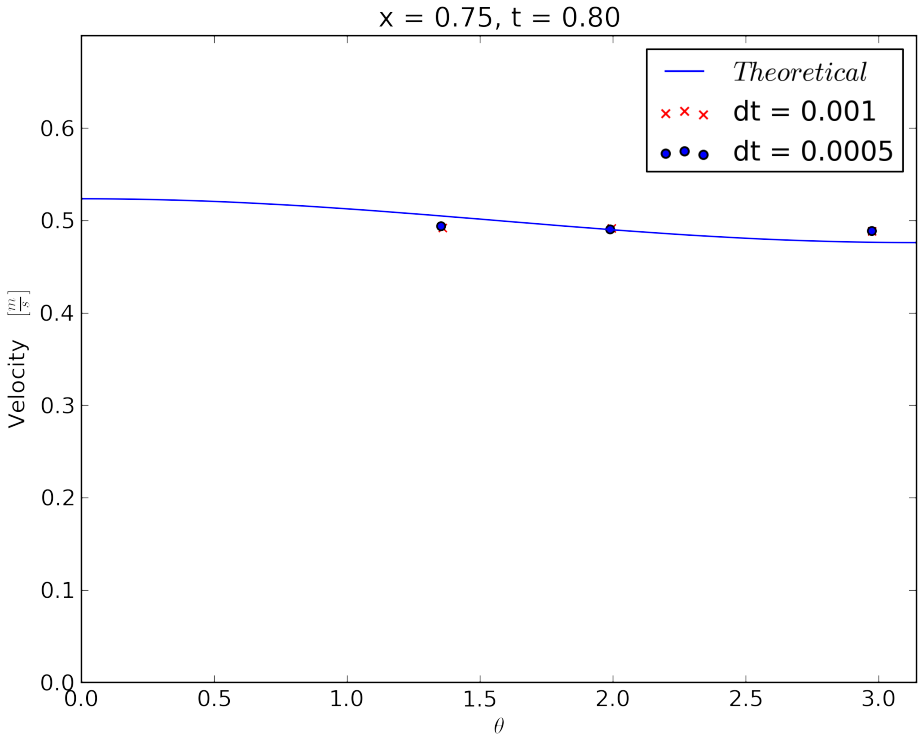

Figure 4.10: X velocity at $x = 0.75$, half cycle

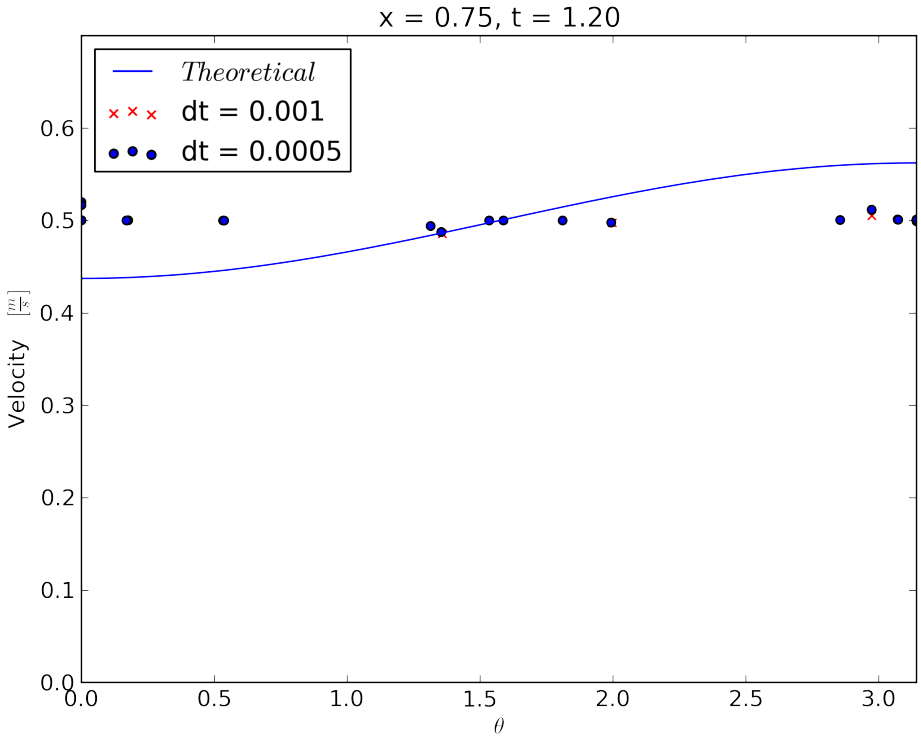

Figure 4.11: X velocity at $x = 0.75$, three-quarters cycle

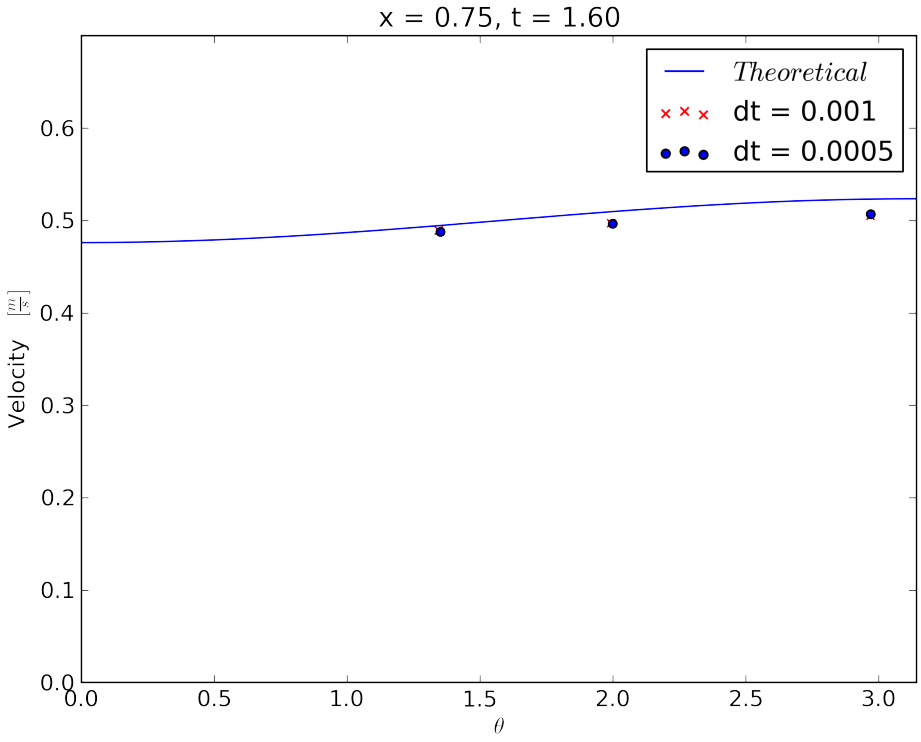

Figure 4.12: X velocity at $x = 0.75$, full cycle

Y velocity

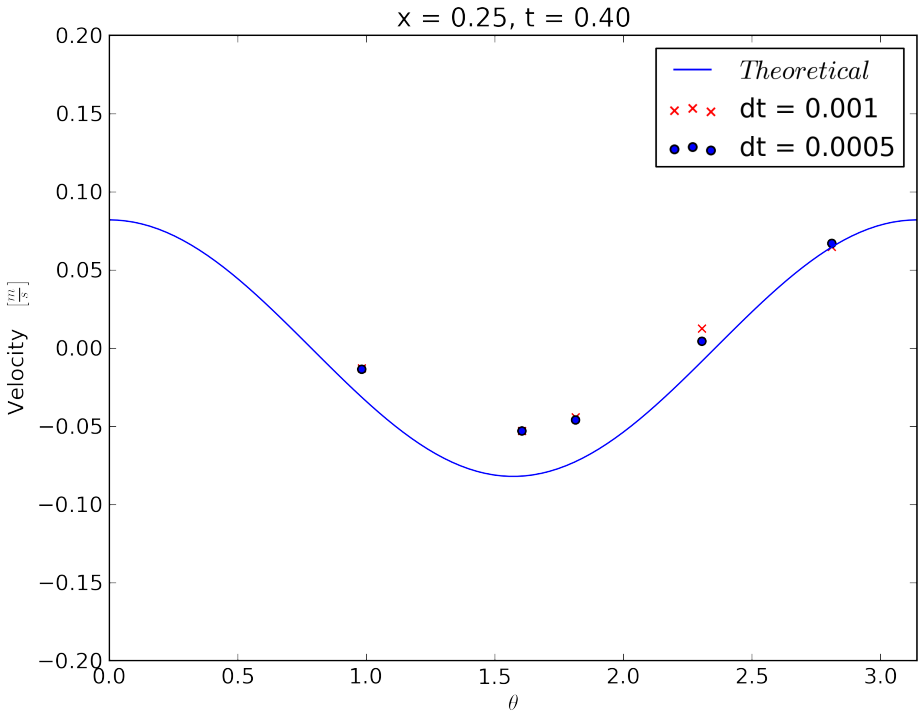

Figure 4.13: Y velocity at $x = 0.25$, quarter cycle

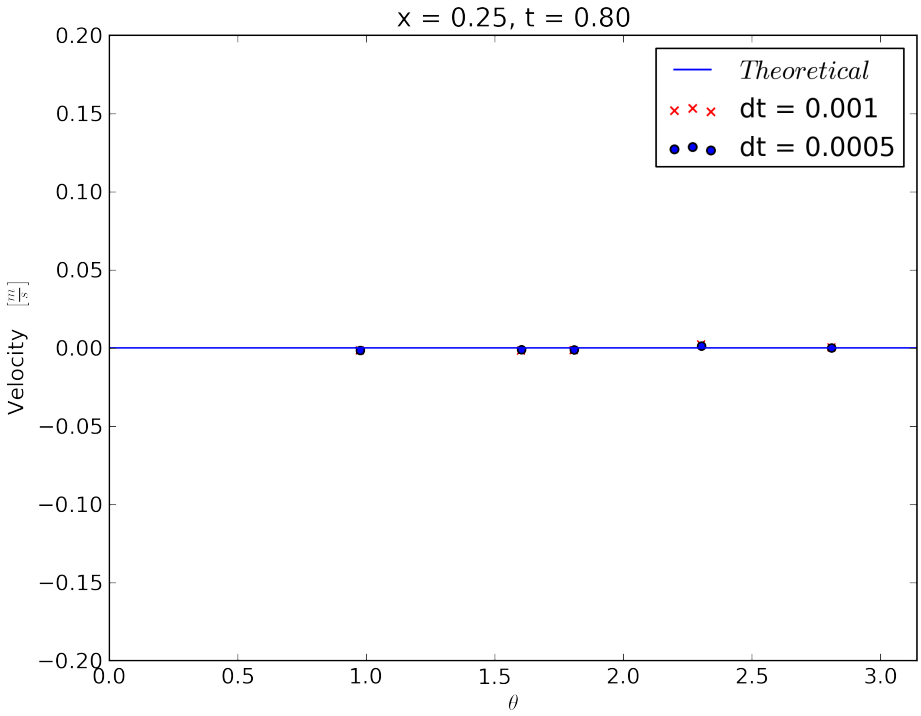

Figure 4.14: Y velocity at $x = 0.25$, half cycle

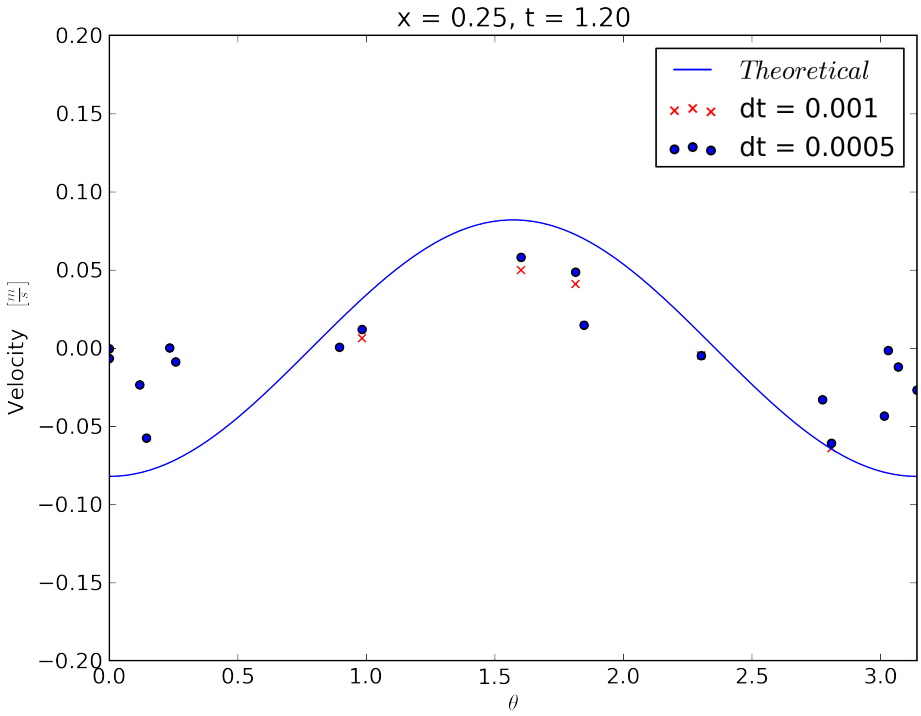

Figure 4.15: Y velocity at $x = 0.25$, three-quarters cycle

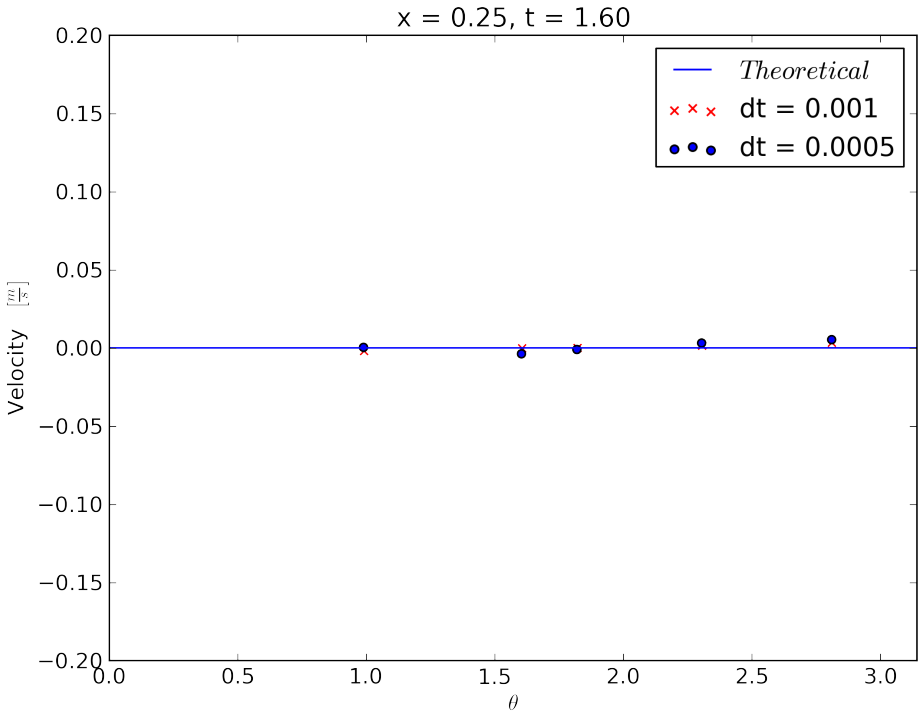

Figure 4.16: Y velocity at $x = 0.25$, full cycle

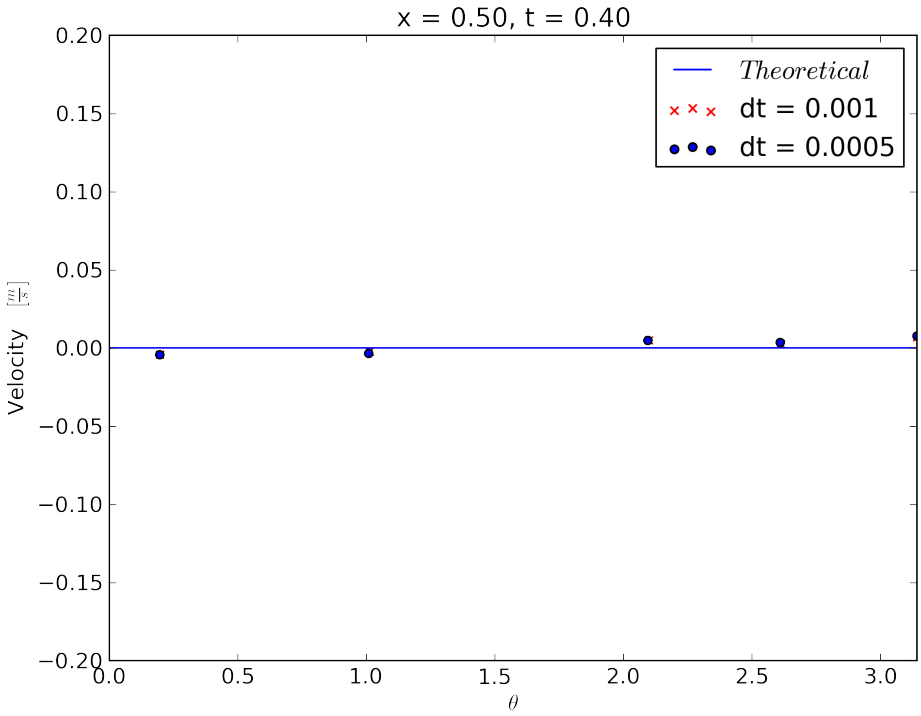

Figure 4.17: Y velocity at $x = 0.50$, quarter cycle

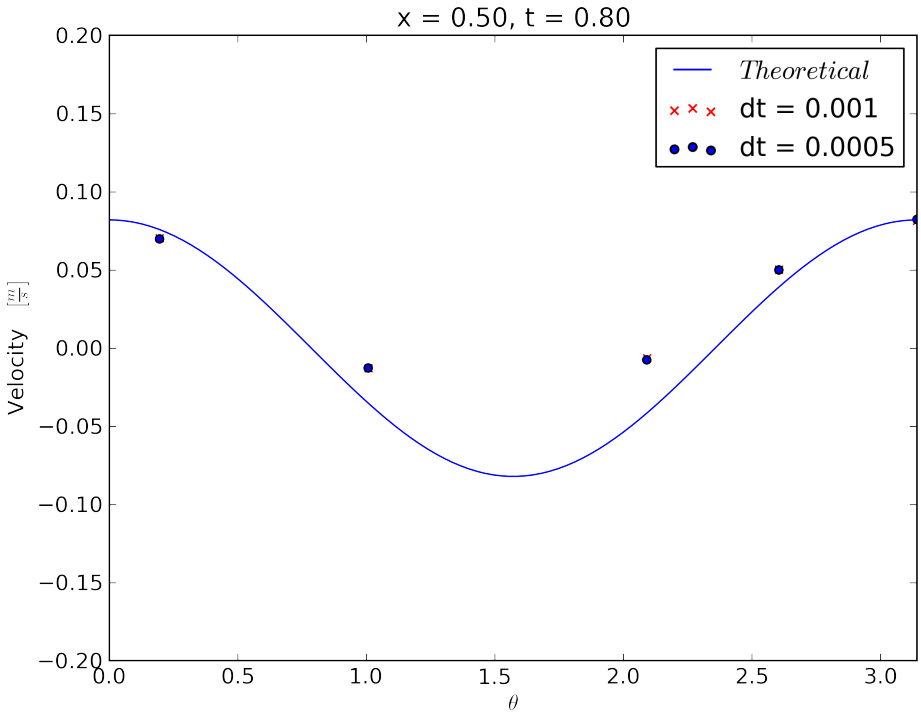

Figure 4.18: Y velocity at $x = 0.50$, half cycle

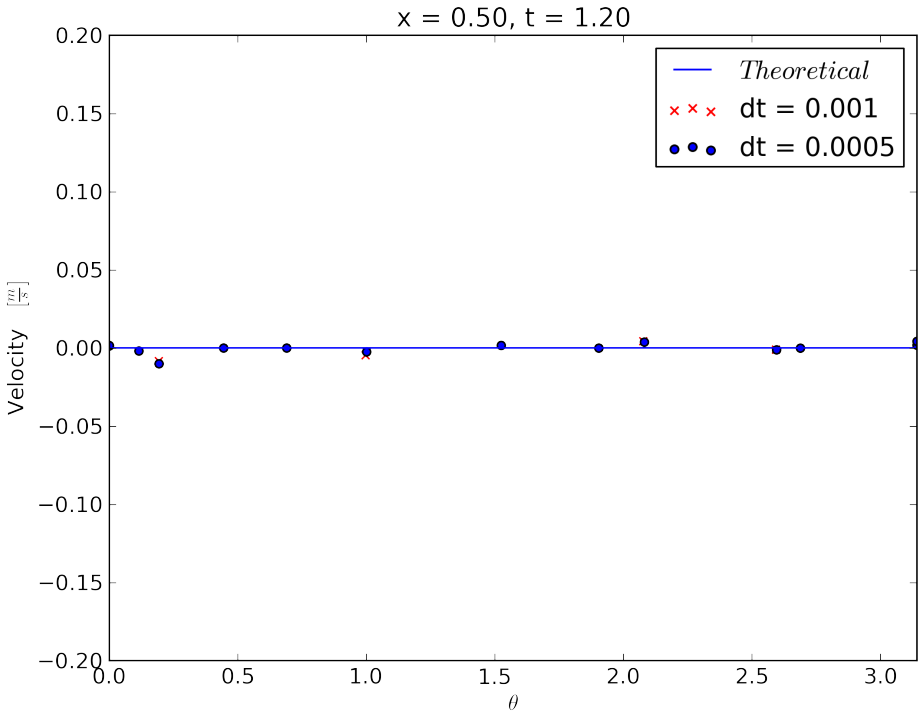

Figure 4.19: Y velocity at $x = 0.50$, three-quarters cycle

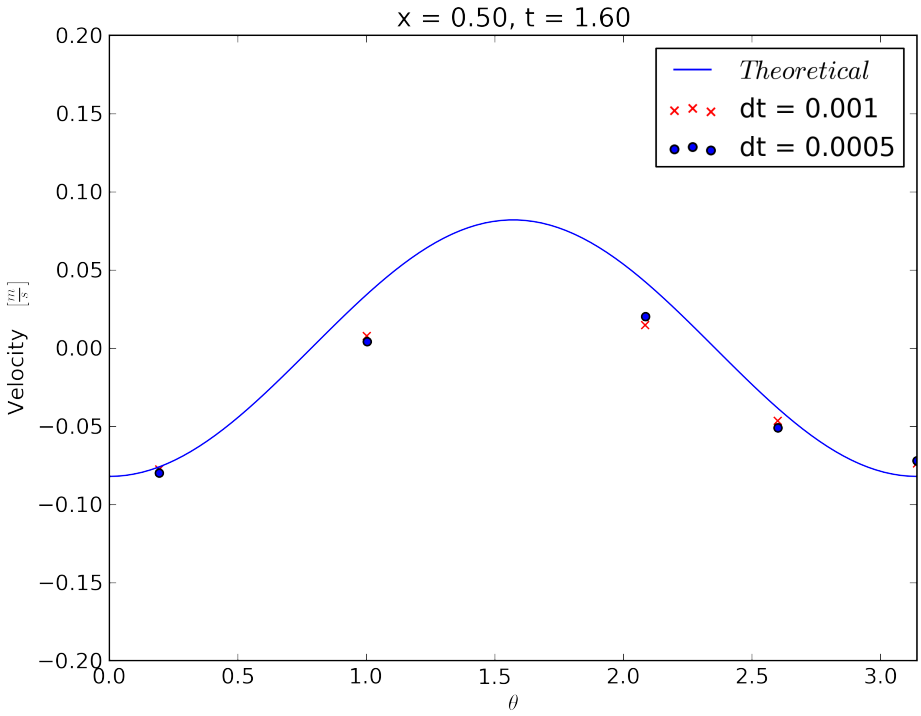

Figure 4.20: Y velocity at $x = 0.50$, full cycle

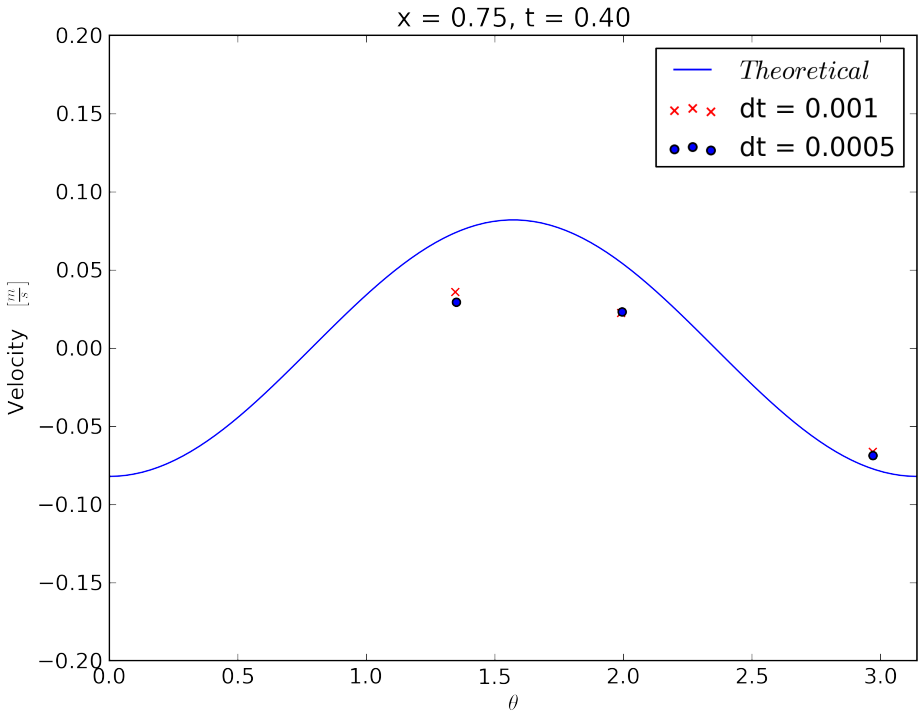

Figure 4.21: Y velocity at $x = 0.75$, quarter cycle

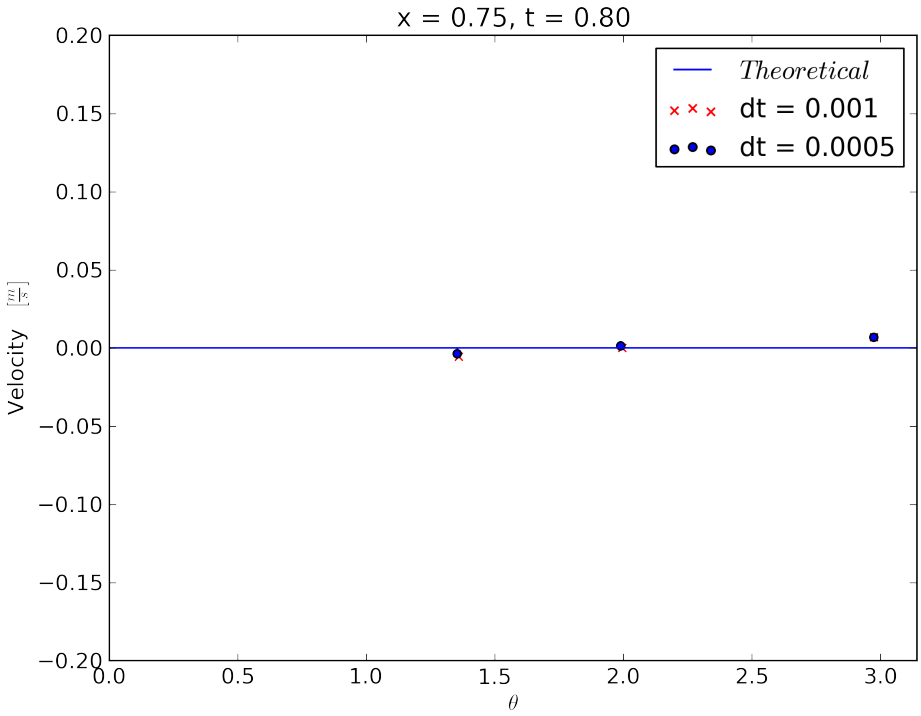

Figure 4.22: Y velocity at $x = 0.75$, half cycle

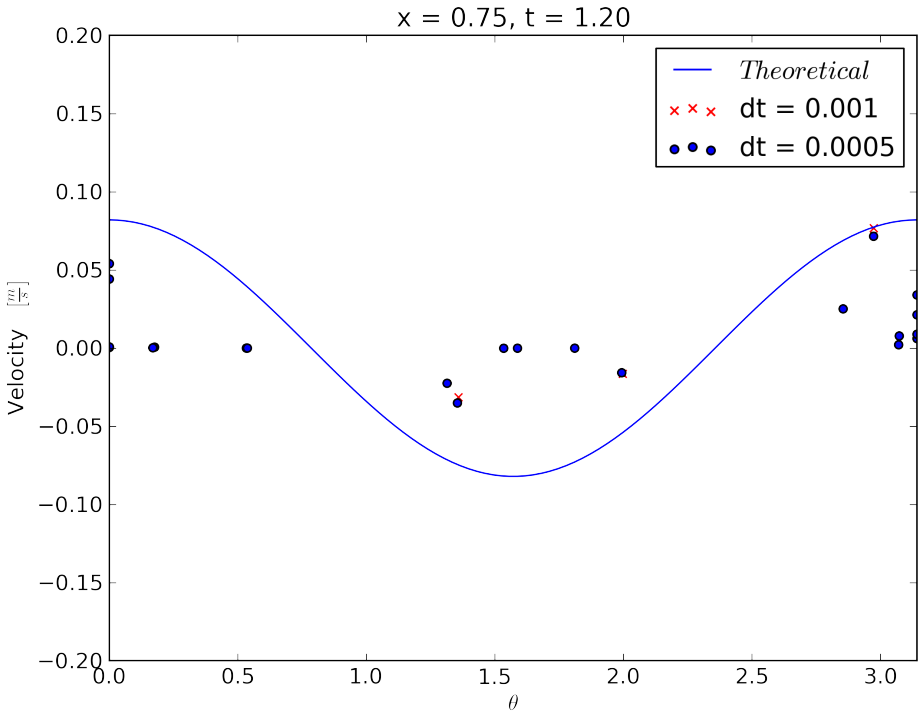

Figure 4.23: Y velocity at $x = 0.75$, three-quarters cycle

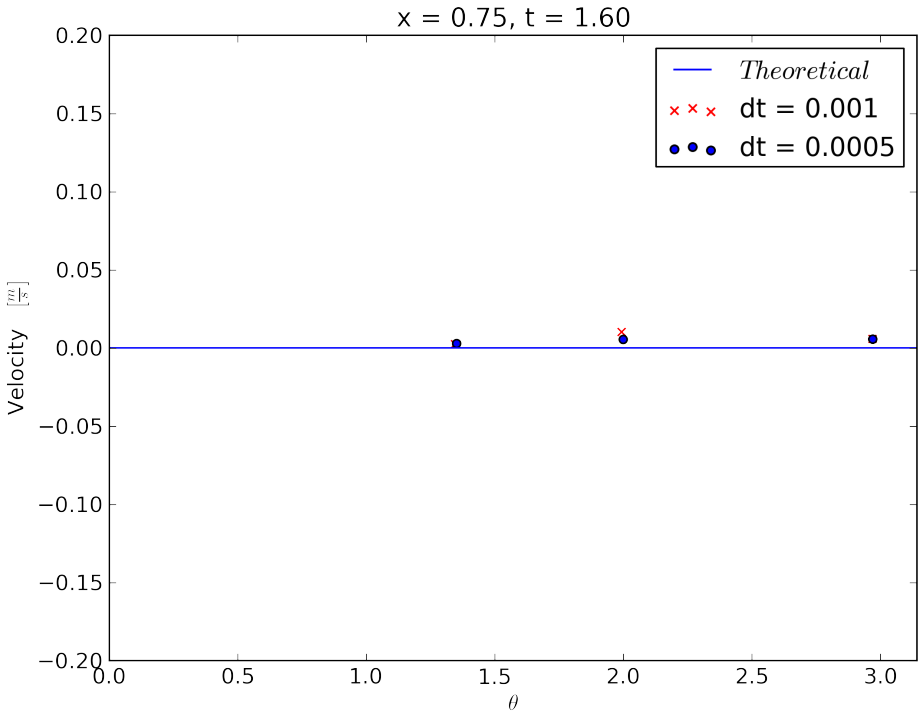

Figure 4.24: Y velocity at $x = 0.75$, full cycle

Z velocity

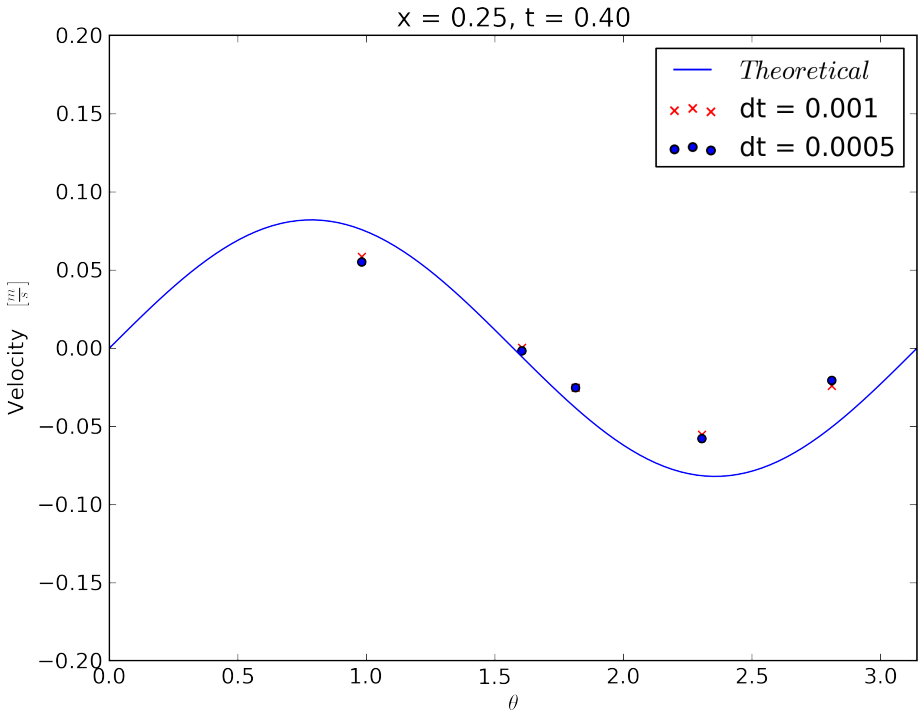

Figure 4.25: Z velocity at $x = 0.25$, quarter cycle

The trends are similar to all of the calculated values. Vorus disregarded the contributions of the y and z components of the velocity when using total velocity in the derivation, and the graphs validate this assumption. In all cases the x velocity is larger than the y and z components by around a factor of 5.

Figures 4.37 through 4.41 show contours of velocity on the symmetry plane around the eel. These are interesting to investigate for any occurrence of downstream vorticity.

Looking at a specific example, Figure 4.39 shows the eel at half of a cycle. Some characteristics are immediately noticeable, such as the stagnation points on the head and tail of the eel. For the most part the flow around the eel is the free stream velocity otherwise, besides an area behind the eel. This low velocity area follows the tail throughout the motion, but doesn't show vorticies happening downstream.

Contours can help give an idea of magnitude, but seeing the same information in vectors can help understand any directional components of the velocity better. Figures 4.42 through 4.46 show velocity vectors in the near field. The focus is on the tail setion where the velocity contours showed lower velocities. Looking at the vectors it is clear that there is some circulation directly behind the tail, effects from the tail hemisphere or the sharp corner connection at the body. There are no vorticies farther downstream however, induced by the motion.

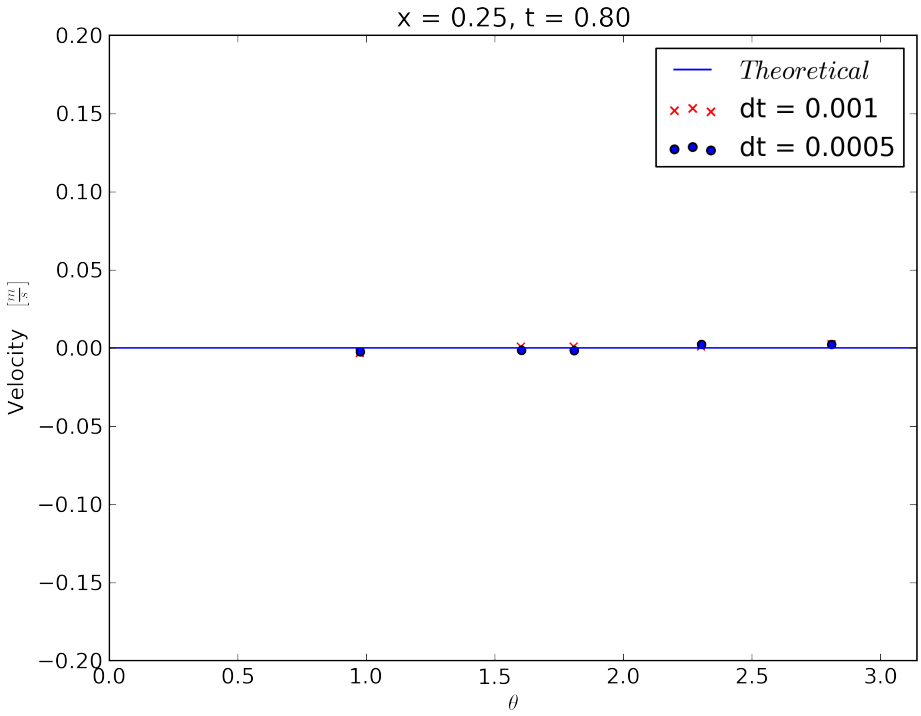

Figure 4.26: Z velocity at $x = 0.25$, half cycle

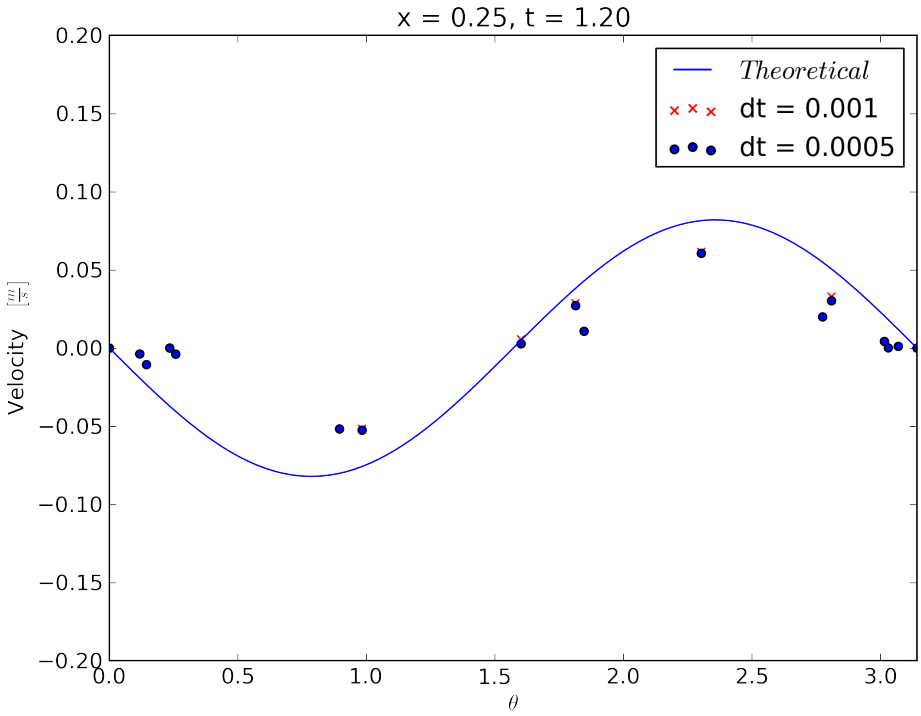

Figure 4.27: Z velocity at $x = 0.25$, three-quarters cycle

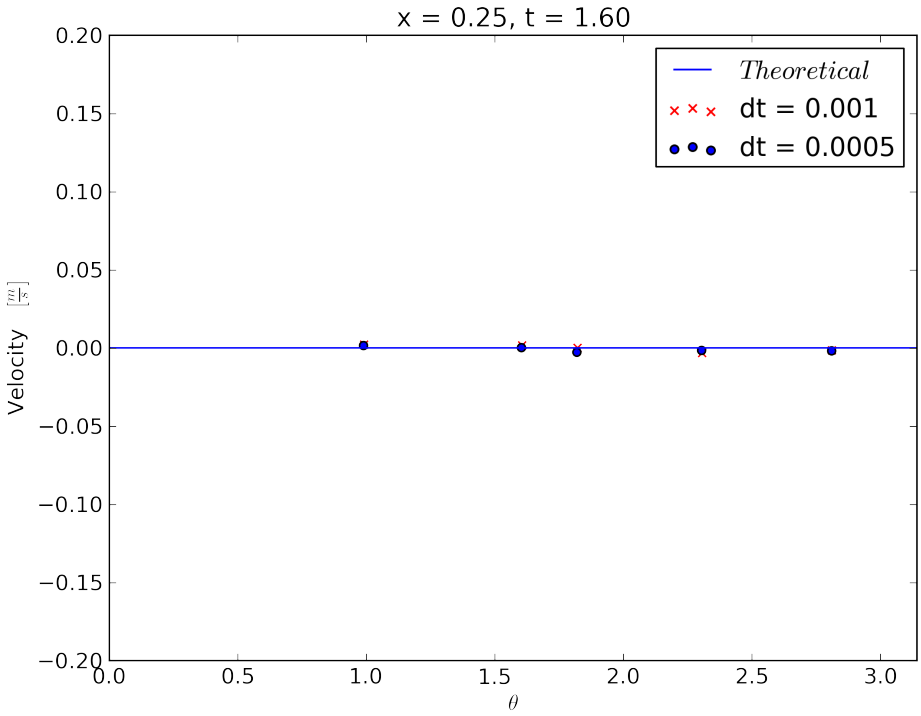

Figure 4.28: Z velocity at $x = 0.25$, full cycle

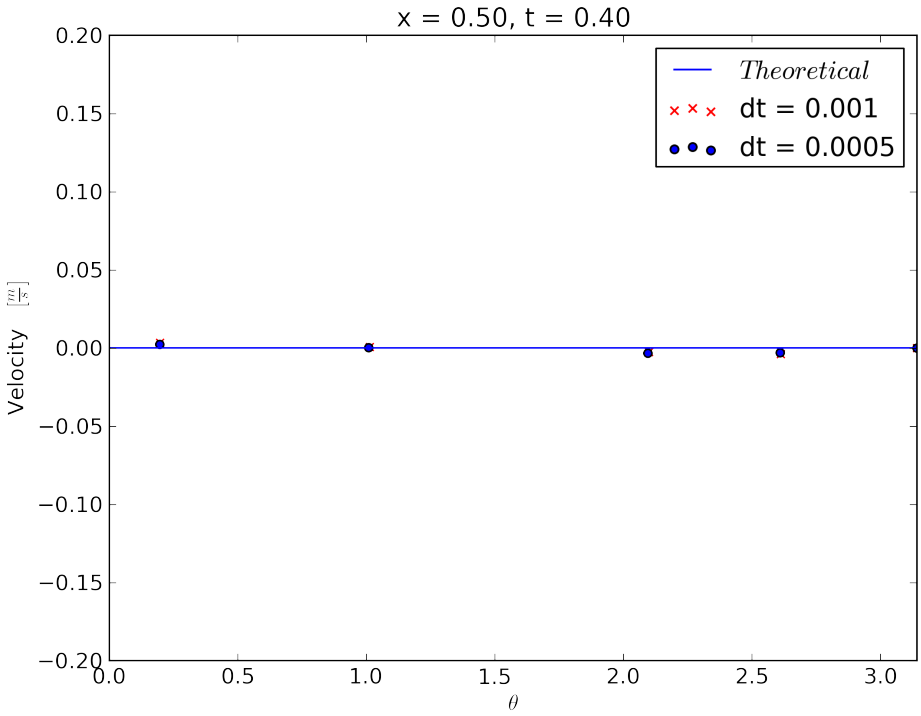

Figure 4.29: Z velocity at $x = 0.50$, quarter cycle

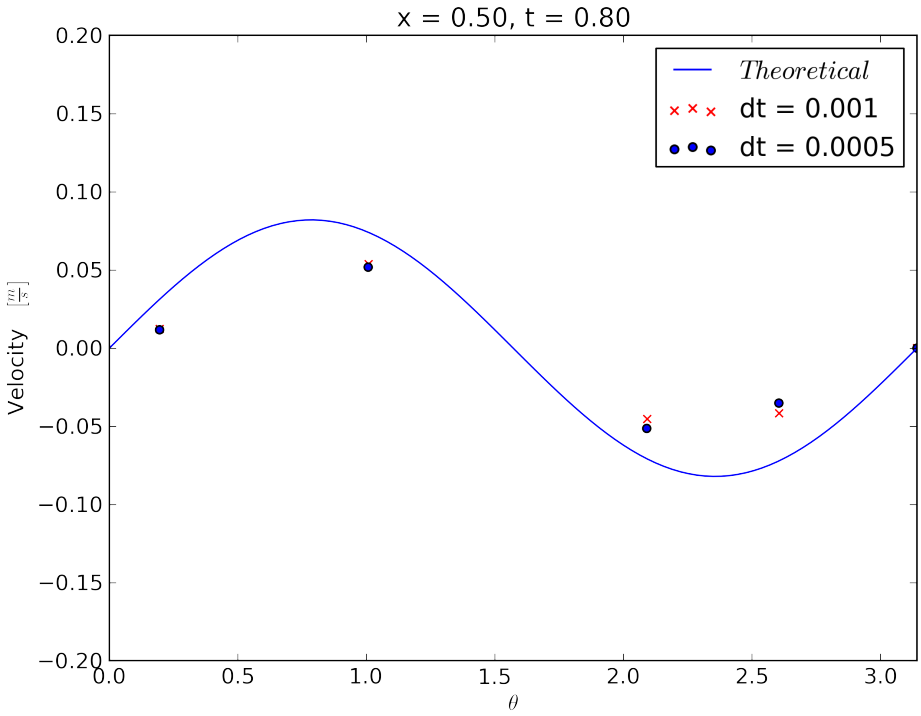

Figure 4.30: Z velocity at $x = 0.50$, half cycle

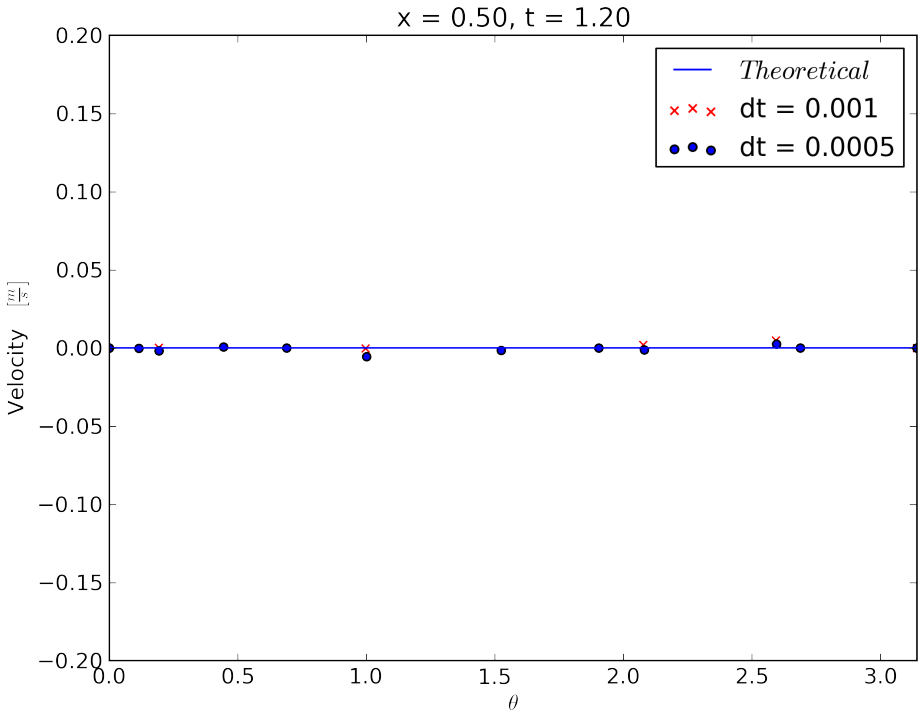

Figure 4.31: Z velocity at $x = 0.50$, three-quarters cycle

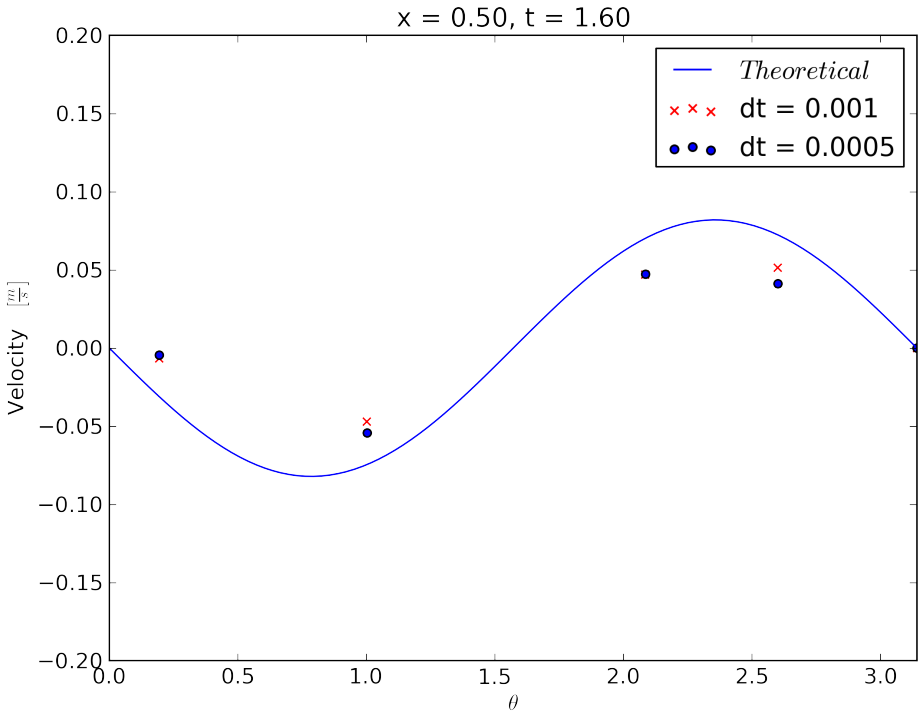

Figure 4.32: Z velocity at $x = 0.50$, full cycle

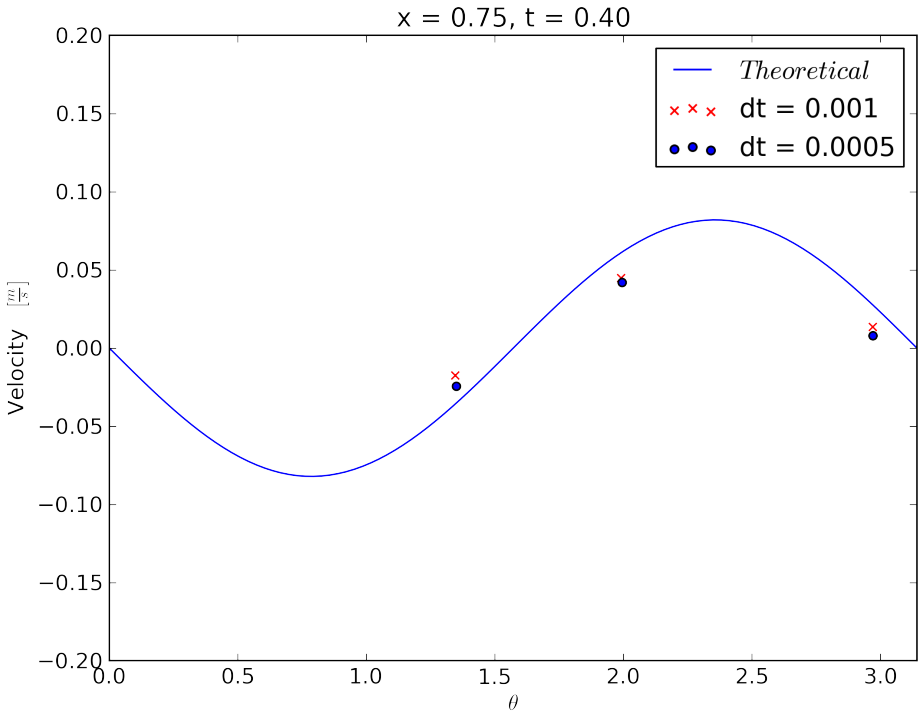

Figure 4.33: Z velocity at $x = 0.75$, quarter cycle

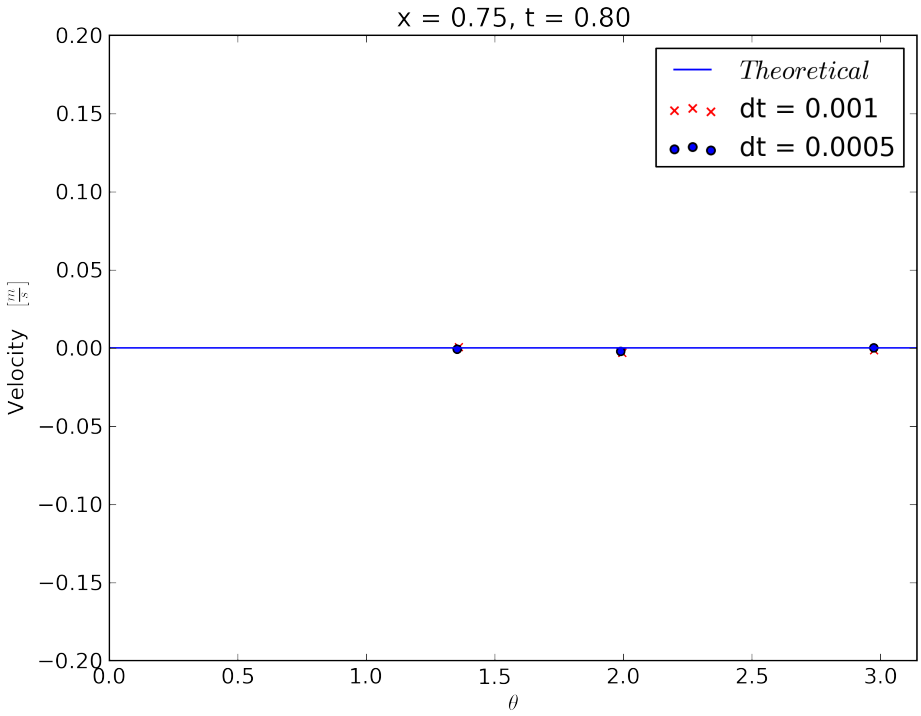

Figure 4.34: Z velocity at $x = 0.75$, half cycle

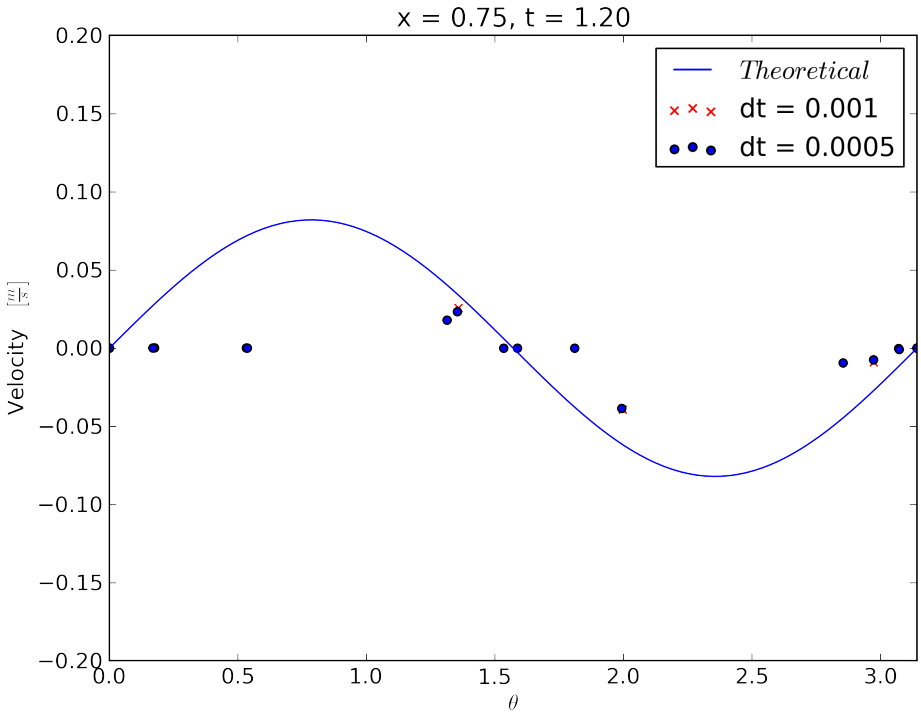

Figure 4.35: Z velocity at $x = 0.75$, three-quarters cycle

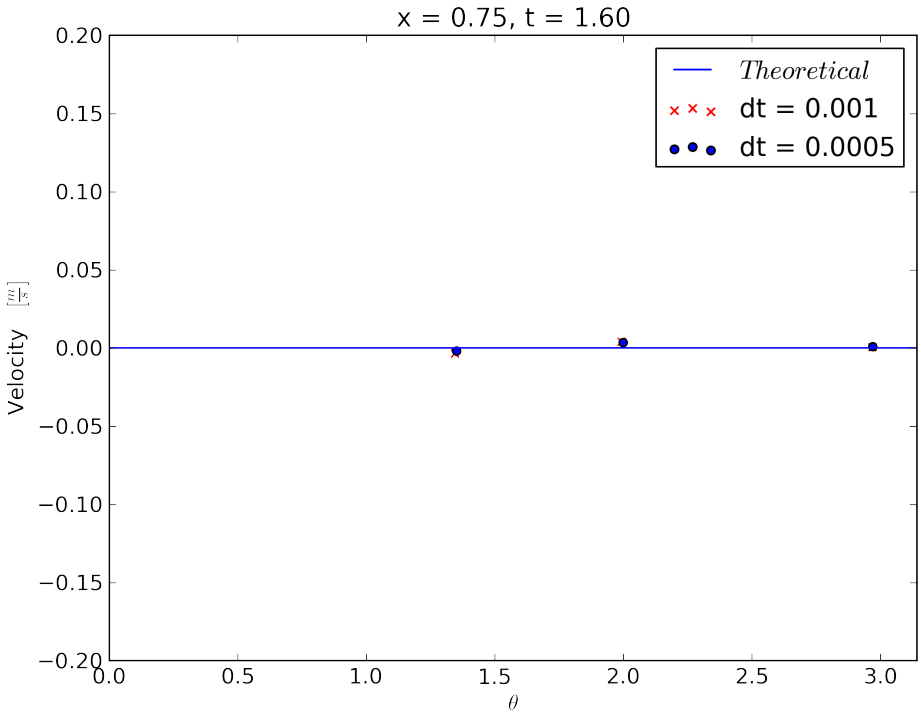

Figure 4.36: Z velocity at $x = 0.75$, full cycle

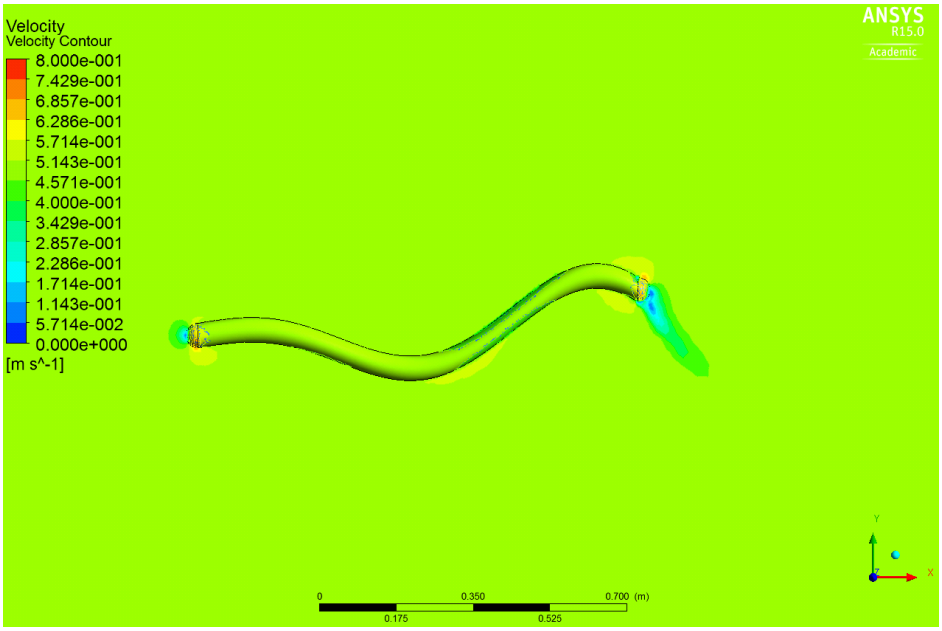

Figure 4.37: Velocity contours at the start of a cycle

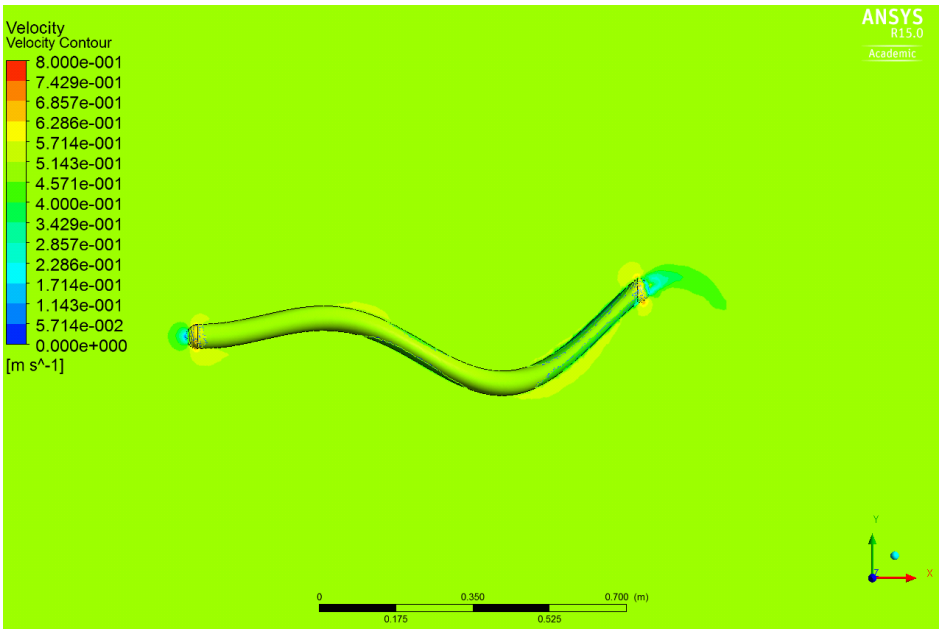

Figure 4.38: Velocity contours at 0.25 cycle

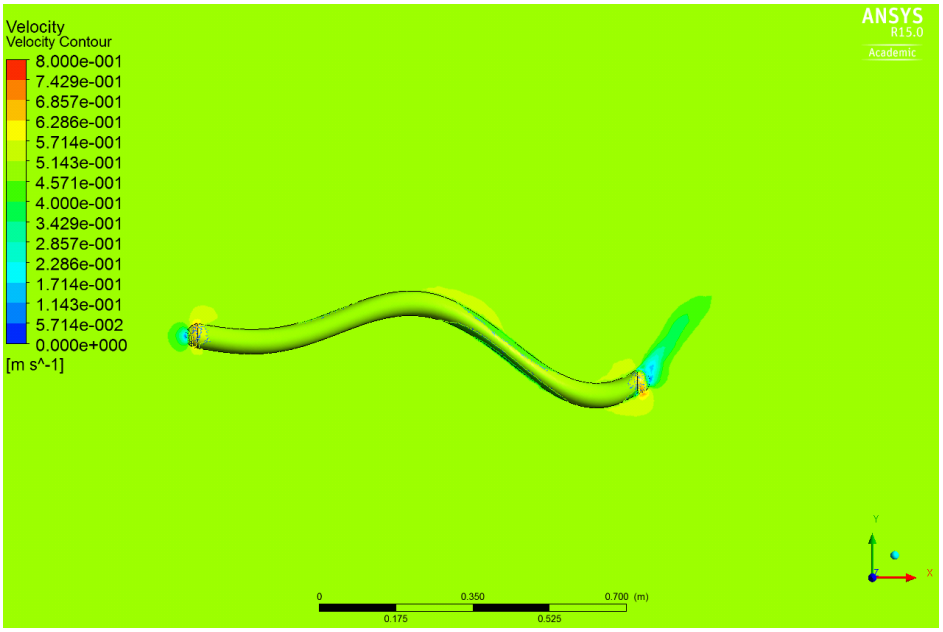

Figure 4.39: Velocity contours at 0.5 cycle

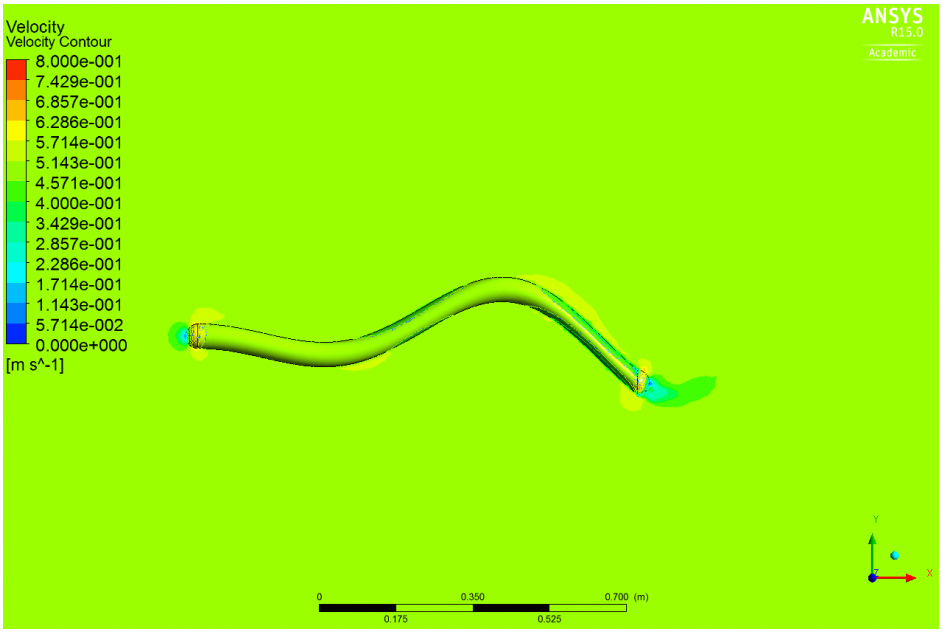

Figure 4.40: Velocity contours at 0.75 cycle

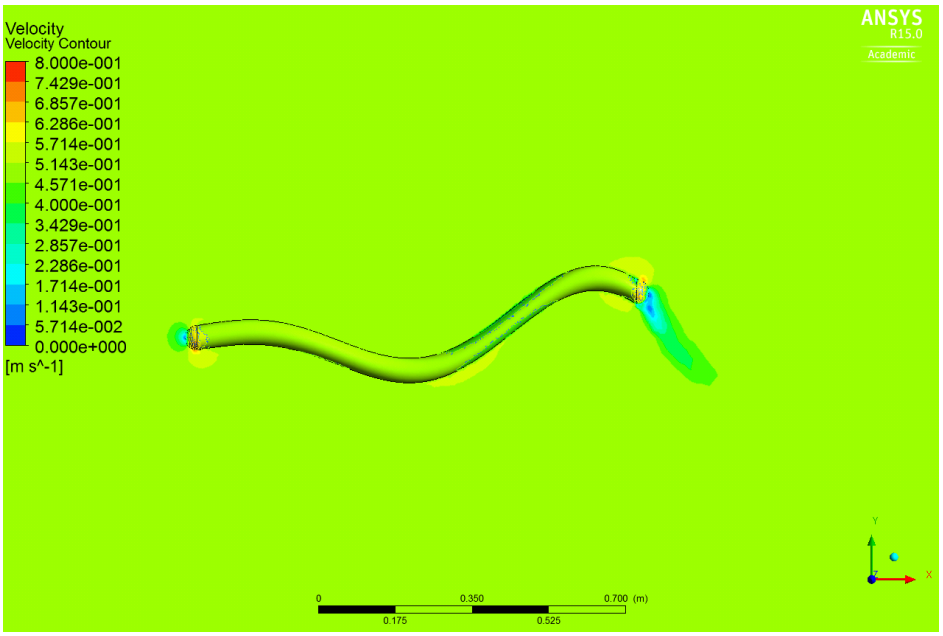

Figure 4.41: Velocity contours at full cycle

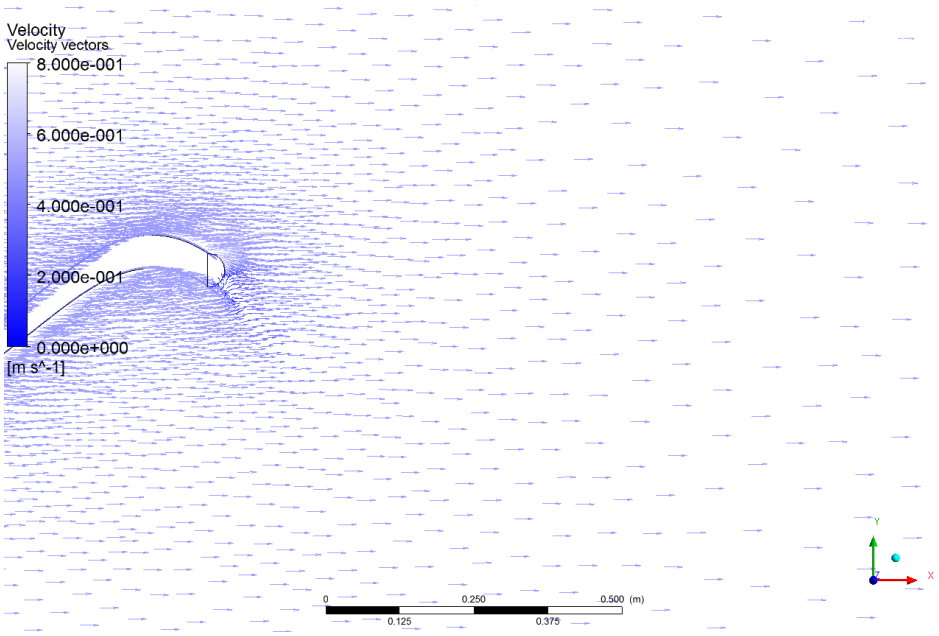

Figure 4.42: Velocity vectors at the start of a cycle

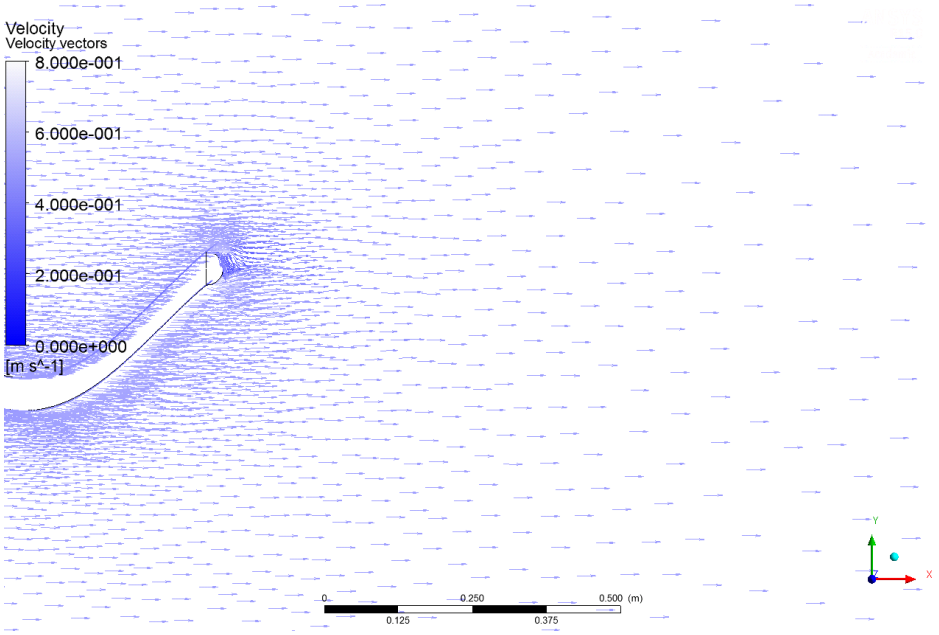

Figure 4.43: Velocity vectors at 0.25 cycle

It is also interesting to see contours of pressure. Figure 4.47 shows the pressure contours at the start of a cycle. There are large pressure spikes similar to the velocity contours on the head and tails near the stagnation points, which is to be expected.

The theoretical thrust coefficient from equation 1.12 can be compared to the calculated drag coefficient from Fluent. Figure 4.48 shows the drag coefficient output from Fluent for both the $dt = 0.001$ and

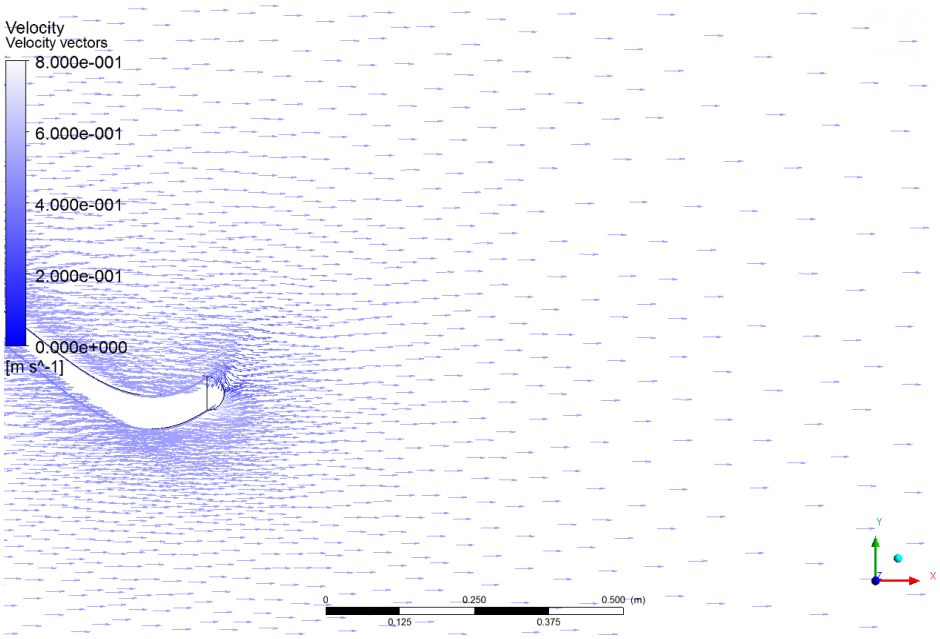

Figure 4.44: Velocity vectors at 0.5 cycle

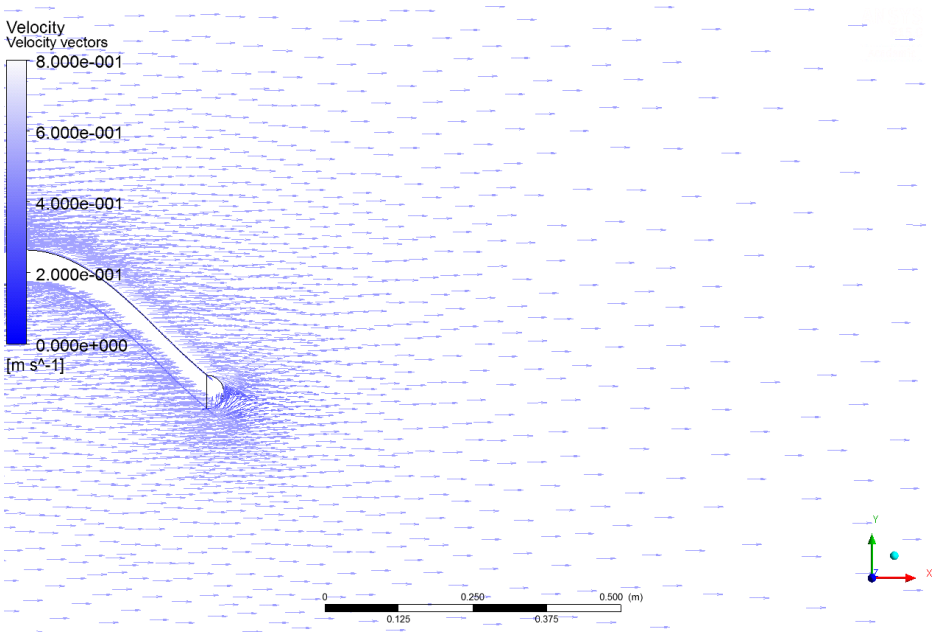

Figure 4.45: Velocity vectors at 0.75 cycle

$dt = 0.0005$ as well as the theoretical component, and the comparisons are both good and bad. The two Fluent outputs compare well to each other, but both are different from the theoretical values (in fact the theoretical curve is multiplied by 10 on the plot). The values between the theoretical and calculated values are different but the general trends are the same.

The main difference is most likely in the calculation of the viscous drag. It is somewhat counter-intuitive

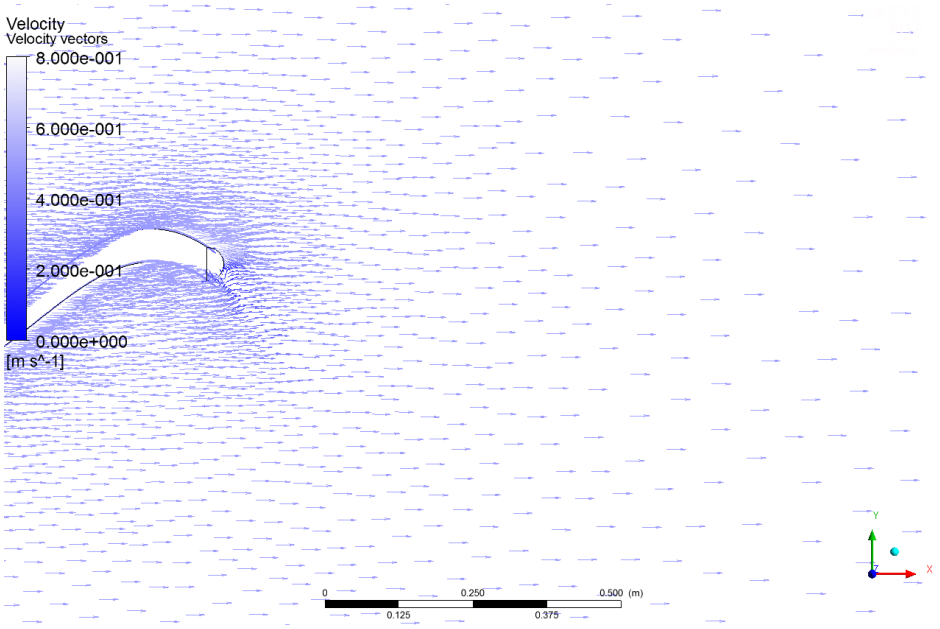

Figure 4.46: Velocity vectors at full cycle

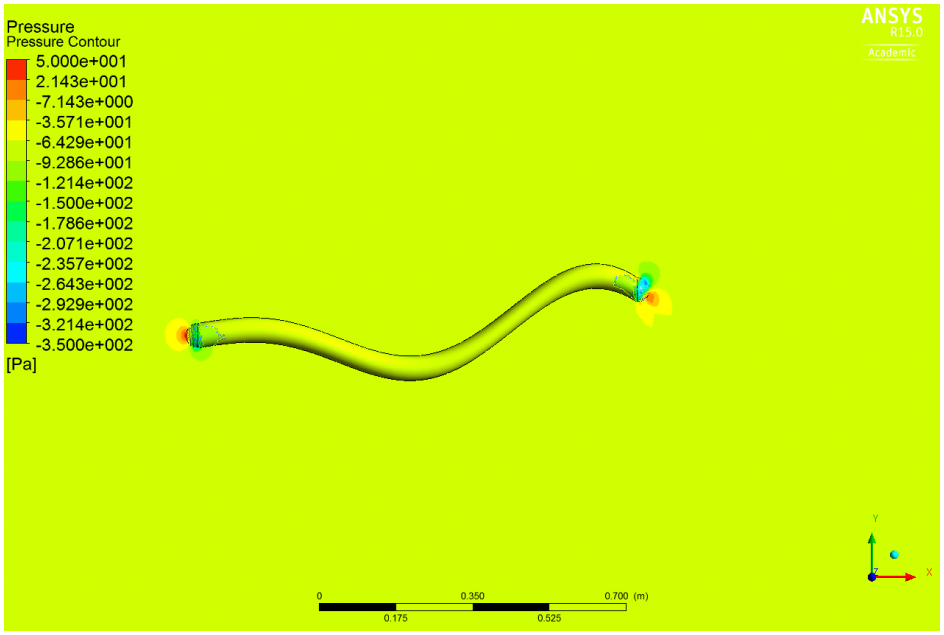

Figure 4.47: Pressure contours at the start of a cycle

that even with an inviscid calculation the viscous drag would be important, but it is a necessary component from equation 1.10. Looking back at equation 1.12 it is apparent that, when 1.12 is integrated along the length, the equation has an offset in the first term and a sinusoidal component in the second term.

Similar to the thrust comparison, any lift output from Fluent can be shown graphically. In this case lift means transverse forces, but theoretically this should be 0 from section 1.1. Figure 4.49 shows the lift output

from Fluent. It is obvious that this is not 0 as the theory described. This result makes sense when looking at the velocity plots

It must be noted that the Fluent data displayed in Figures 4.48 and is just for the eel body, not the head and tail. The theoretical derivation in Vorus and Taravella [16] does not include calculations for the velocity and pressure on the head and tail. It is however apparent looking at Figures 4.37 and 4.47 that there are significant flow characteristics at these locations.

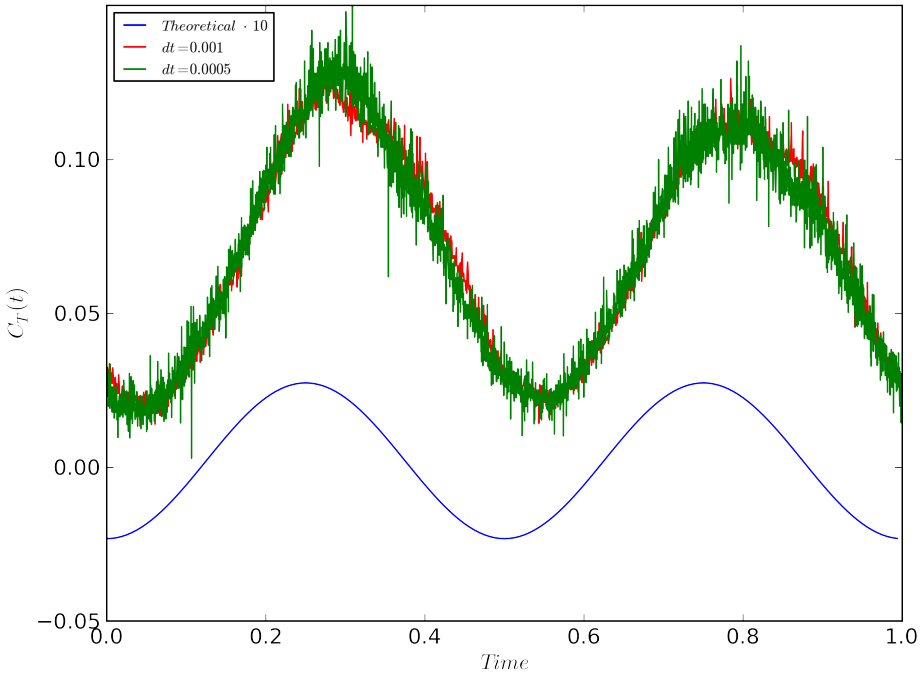

Figure 4.48: Total thrust over time

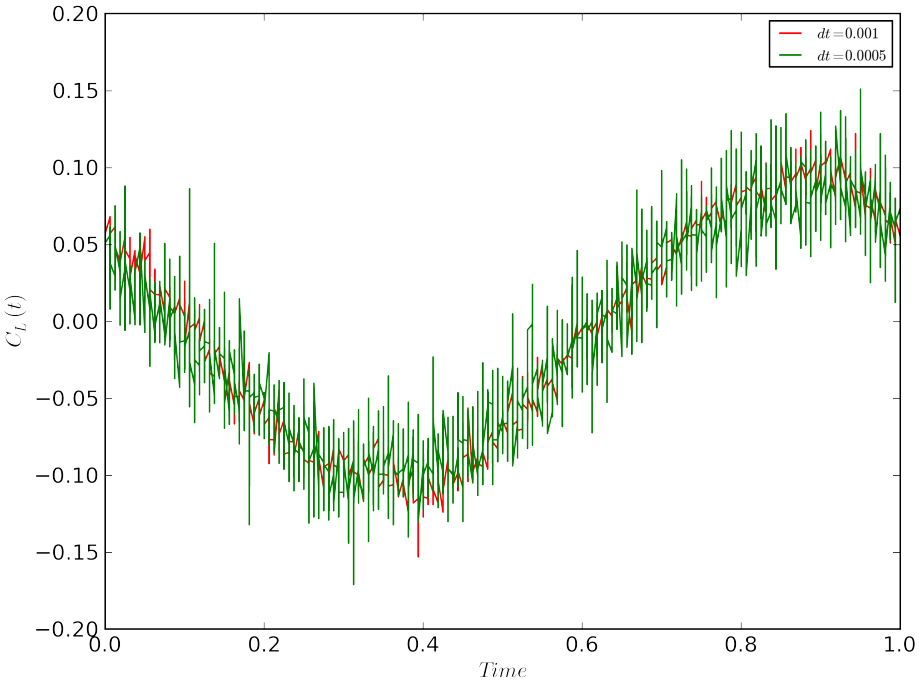

Figure 4.49: Total lift over time

5. Conclusions

5.1 Conclusions

The initial output from Fluent compares well with the expected pressure distributions. Measurable lift and low velocity sections on the plots imply that there is some vortex shedding, most likely due to effects of the tail surface. The thrust comparisons are different, but the implication is that updating the C_T calculation may help lead to a better motion. The methodology also compares well to previously executed analyses for similar problems. From graphs of the velocity it is apparent that there is no induced vorticity downstream, implying that theory has some validity. There needs to be a review of the thrust coefficient calculation, but otherwise the results compare well to the expected values.

5.2 Further Research

As with any research, hindsight offers the opportunity to look back and see where improvements could be made to the work.

One of the first points that could be improved upon is the motion defined by the UDF. As mentioned in section 3.3, the motion used for this work was only a linear displacement of semi-spherical cross-sections, not keeping the cross-sections perpendicular to the centerline of the motion that would be proper anguilliform motion. Work on that juncture would improve the real world applicability of the results.

More generally some improvements can be made to the results by using a more powerful computer for the calculations. Due to licensing restrictions a maximum of four parallel nodes could be used for the calculations. With access to more powerful computational engines more nodes could be added to the mesh to improve accuracy and the calculations could be carried out faster.

Different flow models could also be used, such as a viscous solution as well as turbulent models.

A good validation of the theory would also test the motion in the real world, not just on a computer with CFD. For example, a robot could be built and tested and the results of the CFD could be compared to see if the modeled viscid and turbulent simulations are similar to the actual flow patterns and forces.

Bilbiography

[1] *ANSYS FLUENT User's Guide*. ANSYS Inc., Canonsburg, PA, release 13.0 edition, 2010.

[2] *ANSYS FLUENT Theory Guide*. ANSYS Inc., Canonsburg, PA, release 14.0 edition, 2011.

[3] Frederic Boyer, Mathieu Porez, Alban Leroyer, and Michel Visonneau. Fast dynamics of an eel-like robot, comparisons with navier-stokes simulations. *IEE Transactions on Robotics*, 24(6):1274–1288, 2008.

[4] Jian-Yu Cheng and Georges L. Chahine. Computational hydrodynamics of animal swimming: boundary element method and three- dimensionl vortex wake structure. *Comparative Biochemistry and Physiology - Part A: Molecular and Integrative Physiology*, 131:51–60, 2001.

[5] *FLUENT 6.3 User's Guide*. FLUENT Inc., Lebanon, NH, 2006.

[6] Zhenying Guan, Weimin Gao, Nong Gu, and Saeid Nahavandi. 3d hydrodynamic analysis of a biomimetic robot fish. In *11th International Conference on Control, Automation, Robotics, and Vision*, pages 793–798, Piscataway, NJ, 2010. IEEE.

[7] Justin Wayne Hannon. Image based computational fluid dynamics modeling to simulate fluid flow around a moving fish. Master's thesis, University of Iowa, 2011.

[8] M. J. Lighthill. Note on the swimming of slender fish. *J. Fluid Mechanics*, 9:305–317, 1960.

[9] M. J. Lighthill. Aquatic animal propulsion of high hydromechanical efficieny. *J. Fluid Mechanics*, 44: 265–301, 1970.

[10] Hao Liu, Richard J. Wassersug, and Keiji Kawachi. A computation fluid dynamics study of tadpole swimming. *The Journal of Experimental Biology*, 199:1245–1260, 1996.

[11] John Nicholas Newman. *Marine Hydrodynamics*. The MIT Press, 1977.

[12] Vai Kuong Sin, Wen Yue Deng, and Wei Hang Xiao. Study of motion of flexible eel from the wake of bluff body in a cross flow. In *International Conference of Numerical Analysis and Applied Mathematics*, volume 1479, pages 161–164, 2012.

[13] Jacob Todd. Cfd analysis of the swimming of the semi-infite strip. Master's project, University of New Orleans, 2013.

[14] H. K. Versteeg and W. Malalalsekera. *An Introduction fo Computational Fluid Dynamics: The Finite Volume Method*. Pearson Addison-Wesley, New York, second edition, 2007.

[15] William S. Vorus. Swimming of the semi-infinite strip revisted. *Journal or Engineering Mathematics*, 51:35–55, 2005.

[16] William S. Vorus and Brandon M. Taravella. Anguilliform fish propulsion of highest hydrodynamic efficiency. *Journal of Marine Science and Application*, 10(2):163–174, 2011.

6. Appendix

6.1 UDF code

```
#include "udf.h"
#include "math.h"


/* This program is the udf for the 3D eel motion in Fluent.
   Last update: 140108 ctr
*/



double displacement (double x, double t)
{
        /* Function to calculate the y displacement of an x location
           along the eel. The only inputs are the x position and the time value.

                x  =  x position of the node
                t  =  time for the calculation
                l  =  length of the eel
                G  =  gamma value, calculated in python code
                U  =  non-dimensional advance ratio
                V  =  displacement wave speed
               pi  =  irrational number pi
             xbar  =  non-dimensional x position
             tbar  =  non-dimensional time
                A  =  dummy variable
                B  =  dummy variable
        */

        double l, G, U, V, pi, xbar, tbar, A, B;

        l = 1.0;
        G = 0.10445948;
        U = 0.8;
        V = 0.625;
        pi = 3.141592654;

        xbar = x / l;
        tbar = (V * t) / l;

        A = sin(2. * pi * ((xbar / U) - tbar));
        B = sin(2. * pi * (xbar - tbar));

        return (G * (A - B) * l);
}



DEFINE_GRID_MOTION(eel_theory, domain, dt, time, dtime)
```

```
{

    /* Udf to describe the general motion of the eel in 3D Fluent.
       This is the theoretical motion defined by Vorus at all points,
       therefore this DOES NOT maintain the shape over time.
    */

    Thread *tf = DT_THREAD (dt);
    face_t f;
  Node *node_p;
  double xprev, yprev, hprev, d, h;
  int n;

    /* Variables:

                xprev  =  previous time step x position of the node
                yprev  =  previous time step y position of the node
                hprev  =  previous time centerline displacement at a given x
                    d  =  distance between the centerline and the node
                        h  =  current time centerline displacement
    */


    SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

    begin_f_loop (f, tf)
    {
        f_node_loop (f, tf, n)
        {
            node_p = F_NODE (f, tf, n);

            if (NODE_POS_NEED_UPDATE (node_p))
            {

                NODE_POS_UPDATED (node_p);

                if (CURRENT_TIME > 0)
                {

                    if (NODE_X (node_p) <= 0.0)
                    {
                        // No change, these nodes stay in
                        // the same spot
                        continue;
                    }
                    else if (NODE_X (node_p) > 1.0)
                    {
                        // Previous time position
                        xprev = NODE_X (node_p);
                        yprev = NODE_Y (node_p);

                        // Calculate the previous time
                        // centerline displacement at x = 1.0
                        hprev = displacement(1.0, PREVIOUS_TIME);

                        // Calculate the difference between the
                        // node and the centerline
                        d = yprev - hprev;
```

```
                                        // Calculate the new time
                                        // centerline displacement
                                        h = displacement(1.0, CURRENT_TIME);

                                        // Use the distance calculation to
                                        // give the new node y position
                                        NODE_Y (node_p) = h + d;
                                }
                                else
                                {
                                        // Get the x and y position of the
                                        // node to be moved
                                        xprev = NODE_X (node_p);
                                        yprev = NODE_Y (node_p);

                                        // Calculate the previous time
                                        // centerline displacement
                                        hprev = displacement(xprev, PREVIOUS_TIME);

                                        // Calculate the difference between
                                        // the node and the centerline
                                        d = yprev - hprev;

                                        // Calculate the new time
                                        // centerline displacement
                                        h = displacement(xprev, CURRENT_TIME);

                                        // Use the distance calculation
                                        //  to give the new node y position
                                        NODE_Y (node_p) = h + d;
                                }
                        }
                }
        }
        end_f_loop (f, tf);
}
```

Vita

The author was born in New Orleans, Lousisiana. He was raised in Metairie and received his Bachelor's degree in Naval Architecture and Marine Engineering from the University of New Orleans in 2012. When he was accepted into the graduate program he worked on anguilliform research with Dr. Brandon Taravella.