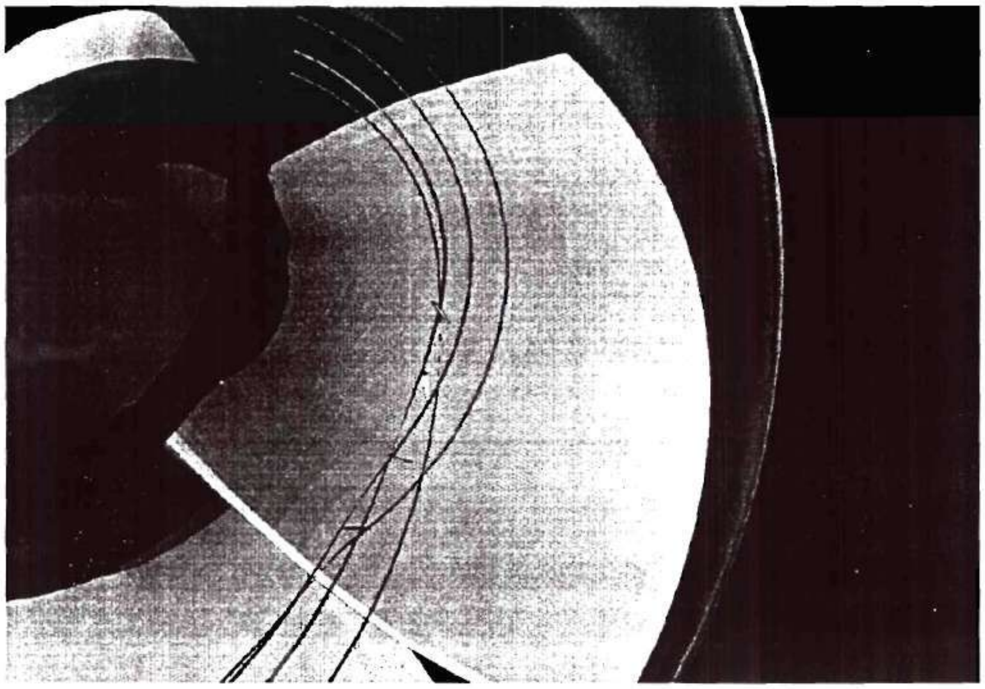
E-20-M94
4,5,6,7

A Numerical Model for Calculating Fish Passage Through Hydraulic Powerplants

by

Y. Ventikos and F. Sotiropoulos

Draft Final Report for Project E-20-M94



Sponsored by
Voith Hydro, Inc.

Environmental Hydraulics and Water Resources Group
School of Civil and Environmental Engineering
Georgia Institute of Technology
Atlanta GA 30332-0355

Table of Contents

1. Executive Summary

This report describes a three-dimensional numerical method for calculating the trajectories of and the loads exerted on fish passing through the various components of a hydraulic powerplant. The model employs a Lagrangian approach for tracking the paths of an arbitrary (user-specified) number of fish, as they are transported by the flow through the powerplant. The flow conditions that drive the motion of the fish correspond to a steady, Reynolds-averaged flowfield obtained via a separate CFD calculation on a fixed (Eulerian) mesh. The geometry of the fish is approximated by a prolate ellipsoid, retaining the basic dimensions and physical properties of the original species (weight, length etc.). It is assumed that the presence of the fish does not alter the precomputed flow field. Moreover, the fish is considered not to respond to the flow (no "free will"), that is, the present method simulates the passage of sedated fish, which are commonly employed in controlled laboratory experiments with real fish. The model accounts for all the important forces that drive the fish motion, can simulate mechanical strike and scrape events and can provide detailed information about all aspects of the flow environment encountered by the fish during its passage.

Since the purpose of this algorithm is the improvement of fish-friendliness of powerplants, the resulting computer tool had to be very efficient and relatively easy to use. This need has guided all the modeling choices that are described in subsequent sections of this report. It should be emphasized, however, that the model has been constructed in a modular form so that its various procedures can be readily enhanced as additional data or more refined models become available.

2. Introduction

Previous work in the area of hydropower plant induced fish mortality has primarily focused on site-specific field experiments that have documented the frequency and type of injuries suffered by passing fish. Such experiments (like those presented in Heisey et al., 1995, 1996; Schoeneman et al., 1991), are usually conducted by releasing a number of tagged fish at chosen locations in the forebay, near the upstream end of the intakes, for various powerplant operating conditions. The tagged fish are subsequently recovered in the tailrace and examined in order to identify and classify the bodily injuries they have suffered. Assuming that enough fish are released per experiment, information can thus be collected on: i) types of injury; ii) correlation of type of injury with release location; and iii) correlation of powerplant operating conditions with overall survivability statistics.

Such experiments have provided most of the available knowledge to date on the impact of hydropower installations on passing fish. For instance, they have helped clarify, and in most cases diss-prove, historical misconceptions regarding widely accepted, "common-sense" correlations between fish survival and turbine operation and design characteristics. Furthermore, they have also led to a first systematic classification, albeit based on after the fact speculation, of possible injury factors (flow induced forces and moments; strike and abrasion on walls and blades; sudden pressure rise or pressure drop; cavitation bubbles collapse; dizziness and disorientation; etc.). Given, however, the manner in which these experiments are conducted and the geometrical complexities of typical powerplants, it is evident that any correlation of the type of injury observed with the actual mechanism that caused it can only be speculative, since:

i) such experiments can neither record the trajectories of fish through the powerplant nor the specific events that led to injury; and

ii) the flow field in a typical powerplant is a very complex, highly three-dimensional, turbulent flow environment, whose physics are still not entirely understood.

Therefore, even if we could design experiments that could yield the precise trajectories of passing fish, we would still need estimates of flow induced loads (pressure and velocity gradients, turbulence fluctuations, etc.) along these trajectories as these, along with mechanical strike and scrape events, are ultimately responsible for injuries. Obtaining such detailed flow measurements along the trajectory of a moving three-dimensional body, however, is very difficult even in model scale, laboratory experiments.

The complexity of the flow environment in the hydraulic components of a powerplant has been a significant roadblock hindering efforts to develop computational models for predicting fish survivability. In recent years, advanced computational fluid dynamics (CFD) methods have been applied with a great deal of success to several complex 3D turbulent flows (Ventikos et al., 1996; Voith Hydro, 1997; Sotiropoulos and Ventikos, 1998; Lin and Sotiropoulos, 1997). These successful applications have demonstrated that CFD offers the only viable alternative today for developing a general numerical model for predicting fish passage through hydropower installations. Assuming that the flow details can be calculated using existing state-of-the-art CFD tools, numerical evaluation of the level of fish-friendliness of a powerplant further requires a computer model capable of tracking fish trajectories from the forebay to the tailrace. For such an approach to be successful, simulations should be carried out for three-dimensional fish-like bodies, closely resembling the geometrical and physical characteristics of the species under consideration, so that the distribution of flow-induced forces and moments acting on the fish body can be computed in detail everywhere along the calculated trajectories. Species-specific biological information could subsequently translate this data into estimates of injury and/or mortality rates. Existing fish passage models, however, are mainly based on geometrical assumptions and can only give gross estimates about the probability of fish impact on the turbine blades (Voith Hydro, 1997). These models have built into them little or no detailed flow physics and can not be used to predict mortality which is the combination of numerous poorly understood phenomena.

The objective of this work is to develop an innovative numerical model for simulating, in a realistic manner, the passage of fish through various components of hydropower installations over the entire range of operating conditions. The input for such a model consists of:

i) the geometrical and physical characteristics of the fish species under consideration; and

ii) a complete three-dimensional solution (in terms of mean velocity components, pressure, and turbulence quantities) of the flowfield through the subsystem in which fish trajectories need to be predicted.

The model predicts three-dimensional fish trajectories, thus directly yielding information for mechanical strike and potential for abrasion. Furthermore, at every point along the predicted trajectory, the distributions of the various flow quantities (pressure, shear stresses, turbulence fluctuations, etc.) on the fish body may be readily calculated via interpolation from the surrounding flowfield. The interpolated flow quantities are used to estimate, among others, pressure variations on crucial fish body parts (head, bladder, etc.), direction and magnitude of flow induced forces on fish, dizziness effects due to intense spinning, potential for cavitation-related damage, etc..

In what follows, we begin with a review of previous studies on fish passage in hydropower installations. Subsequently, we describe the overall architecture and the various components of the model and discuss a small set of representative results in typical powerplant components. A detailed description of the computer code, along with a Users Manual is presented in the Appendix.

3. Review of Previous Work

In this section, we briefly review available in the literature studies aimed at elucidating the causes of powerplant-induced fish injuries and mortality. It is important to emphasize that although anadromous species, like salmon and American shad, have to migrate from freshwater to marine habitat and back during their life cycles, our emphasis herein is exclusively on downstream migration. This is because the upstream journey is usually facilitated by the use of ladders, lifts and other mechanical aids (Cada and Sale, 1993), that have been found to induce practically no mortality to the species. On the other hand, similar efforts to collect downstream-travelling fish upstream of the powerplant and appropriately direct them, with the use of similar mechanical devices (screens, water-blowers etc.), to bypass the turbines have yielded inconsistent, site-specific, results--being very successful in some powerplants but not in others. Moreover, there have been some cases studies reported in the literature in which the survivability of fish passing through the turbines was observed to be identical to that of fish bypassing the powerplant (Heisey et al., 1996). For these reasons, and due to the fact that most of the powerplants in the US are due for re-licensing and retrofitting, most recent work in this area has focused on understanding fish passage through the powerplant and developing innovative remedies for eliminating mortality.

The vast majority of previous studies have relied on field and laboratory experiments. As already discussed in a previous section, the former experiments are by nature site-specific and their objective is to correlate, in a global sense, the frequency and type of injuries suffered by passing fish in terms of upstream release location and powerplant operating conditions. Laboratory experiments, on the other hand, are typically designed to study a particular mortality mechanism. For that reason they are specifically tailored to reproduce that mechanism in the laboratory and quantify critical, for inducing injury and/or mortality, threshold loads on fish. As the subsequent discussion will show, some mortality mechanisms have received more attention than others. For example, there is a significant amount of work on the effects of pressure and pressure variation, whereas very little has been done to study possible effects of turbulence induced mortality. Such disproportionate focus should by no means be interpreted as a reflection of the relative importance of various mortality mechanisms. It should be rather attributed to the difficulties in conducting meaningful experiments for some mechanisms (like turbulence induced mortality) and to the fact that the underlying fluid mechanics phenomena associated with some mechanisms are not entirely understood.

The subsequent discussion is not intended to be a comprehensive literature review, as such a task has been already undertaken successfully in a number of recent studies (see, for example, Cada and Sale (1993) for a thorough review of recent work in the area). It is, however, intended to be a critical discussion with emphasis on examining and evaluating the extent to which existing data sets can provide us with the information necessary for facilitating our modeling efforts.

We choose to classify various laboratory experiments based on the specific injury mechanism they examined. The following mechanisms are considered: i) pressure and pressure variation effects; ii) cavitation; ii) shearing forces; iii) turbulence; and iv) mechanical impact.

3.1 Pressure and pressure variation effects

Fish travelling through the powerplant continuously encounter environments of changing ambient pressure. Given the very short residency time within the plant, it is important to investigate not only the impact of the absolute pressure level on the fish health but also the possible effects of the rate at which pressure changes along fish trajectories. A number of experiments, with a variety of species, have shown that the typical pressure levels associated with the various subsystems of hydraulic powerplants do not induce any significant mortality, provided that the fish have the opportunity to gradually acclimate to these pressure levels, (Foye and Scott, 1965). In essence these results suggest that compression and decompression at slow rates is non-fatal.

The same, however, does not hold for higher rates of pressure change (especially decompression). In order to understand the reasons that cause high mortality rates in rapidly decompressing fish, let us briefly comment on an important aspect of fish physiology. Depending on how the swim bladder of fish exchanges air with the environment, fish species can be classified into two broad categories: i) physoclistous species (like perch, bass, etc.), which have a closed bladder that changes its air content through diffusion to the blood system of the fish; ii) physostomous fish (like salmon, catfish etc), which have a small duct connecting the bladder with the environment (usually through the mouth cavity). There is a great difference at the response speed of these two systems to pressure variations. Physostomous fish react in a matter of seconds, whereas for physoclistous fish the process may take a few hours (Lagler et al., 1962). It is, therefore, evident that physoclistous fish are much more susceptible to rapid pressure variations than physostomous fish, a trend which has been verified in a number of laboratory experiments (Tsvetkov et al., 1972; Turnpenny et al., 1992; Feathers and Knable, 1983). It should be pointed out, however, that extremely large temporal changes of pressure can also harm physostomous fish. Since, therefore, typical passage time through high flow turbines is very small (order of 1 second), the total pressure drop across the dam plays a crucial role in determining this kind of mortality. Pressure sensitivity experiments have typically been carried out in closed pressurized chambers in which the water pressure can be accuratelly controlled via mechanical means, (Feathers and Knable, 1983; Tsvetkov et al., 1972; Turnpenny et al., 1992). More unusual tests, in which the pressure variations were created by underwater explosions, have also been conducted (Traxler et al., 1993). In both approaches, fish that are acclimated to a specified pressure, usually near the atmospheric pressure, experience a sudden pressure change. The resulting temporal pressure variations experienced by the fish were then recorded and correlated with observed injury and mortality rates. It was thus shown that pressure drops more rapid than about 90 KPa/sec cause significant mortality rates.

It should be noted that the relative ease that such experiments can be set up and conducted has led to an extensive data set, perhaps the most comprehensive among those available for other mortality sources. Despite the large body of data, however, very few of these experiments were systematic and detailed enough to provide information that is readily usable within a modeling framework.

3.2 Cavitation

Cavitation is probably the most complex and intriguing fluid mechanics phenomenon occurring in hydraulic turbomachinery. When the water pressure drops below a critical threshold (usually at the suction side of blades, blade tips or high velocity regions) pockets of vapor are created in the fluid, providing water temperature and purity are favorable (Knapp et al., 1970; Young, 1989). When these vapor bubbles are transported by the flow to areas of higher pressure, they collapse and create shock waves, of originally spherical shape, that propagate through the water. These shock waves are extremely intense as they can reach several hundred atmospheres in the immediate vicinity of the collapse location. When a fish happens to be in the proximity of such an event, hemorrhaging of the eyes and gills of moderate to fatal severity can occur.

The impact of cavitation on passing fish has been studied experimentally in small pressure controlled cavitation chambers (Muir, 1959; Turnpenny et al., 1992). It is of course very difficult to reproduce the exact conditions existing in the turbine, because of scale effects as well as the fact that, as mentioned before, the phenomenon is very complex and not fully understood or simulated numerically. Although very little experimental evidence is available (Turnpenny et al., 1992), it is reasonable to assume that since metal elements of the hydraulic machinery can sustain considerable cavitation-induced damage, collapsing cavitation bubbles could inflict potentially serious injuries on passing fish. A concrete correlation, however, between cavitation and fish injuries has yet to established experimentally.

One approach that can be used to alleviate the intensity of shock wave induced by a collapsing cavity is to introduce air bubbles into the water. Under certain conditions, the resulting air pockets have been shown to provide a cushion that can absorp some of the energy released by the cavity, thus, acting like a shock-wave damping mechanism (Chanson, 1989). These findings led Cada and Coutant (1997) to speculate that air injection, an approach already employed in autoventing hydroturbines for enhancing the dissolved oxygen concentration of the tailwater, could also be suitable for reducing cavitation-related injuries. Further experimental and numerical modeling studies are needed, however, before more specific conclusions can be reached. These studies must focus on: i) clarifying the physics of cavitation in hydroturbines; ii) documenting the nature and quantifying the extent of cavitation-induced injuries; and iii) evaluating the overall effectiveness and impact on turbine efficiency of remedies based on air-injection strategies.

3.3 Shearing loads

Before proceeding to review experiments that focused on shear-induced injuries and mortality, let us briefly review some fundamental fluid mechanics concepts regarding the definition of the shear-stress concept in a fluid. This brief review will be useful in our subsequent discussion as it will help us evaluate the completeness and usefulness of existing experiments as well as identify specific needs for future experiments.

The flowfield in most subsystems of a hydropower plant is very complex, highly three-dimensional and unsteady. The level of spatial complexity of a flowfield can be quantified in terms of the gradients of the three velocity components (u, v, w) along each spatial direction (x, y, z). For a three-dimensional flow, these gradients comprise a 3x3 velocity-gradient tensor, defined as follows:

$$\nabla \vec{V} = \begin{bmatrix} \partial u/\partial x & \partial u/\partial y & \partial u/\partial z \\ \partial v/\partial x & \partial v/\partial y & \partial v/\partial z \\ \partial w/\partial x & \partial w/\partial y & \partial w/\partial z \end{bmatrix} \tag{1}$$

As a general rule, the complexity of a flowfield is directly proportional to the number of non-zero components of the above tensor. For example, for parallel Couette flow (flow driven by two flat belts moving at different speeds), the simplest viscous flowfield, the velocity gradient tensor has only non-zero component, $\partial u/\partial y$. Within the powerplant, on the other hand, there are no regions where some components of the velocity gradient tensor are identically zero, although in certain areas some may be considerably smaller than others. For instance, within the intake the spatial derivatives of u, the velocity component along the predominant flow direction, should be expected to de significantly greater than the derivatives of the other velocity components. Further downstream, however, the geometrical complexities of the distributor and the wicket gates, the rotation of the runner, and the complex curvatures and area expansion within the draft-tube make all velocity derivatives non-zero and of comparable relative magnitude.

For a viscous fluid, the velocity gradient tensor produces a shear-stress field which is also quantified in terms of a 3x3 tensor, the so-called shear-stress tensor. For a Newtonian fluid of viscosity μ, this tensor is symmetric and is expressed in terms of the velocity gradients as follows:

$$\bar{\bar{\tau}} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \tau_{yy} & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \tau_{zz} \end{bmatrix} \equiv \mu\left(\nabla\vec{V} + \nabla\vec{V}^{T}\right) = \mu\begin{bmatrix} 2\partial u/\partial x & \partial u/\partial y + \partial v/\partial x & \partial u/\partial z + \partial w/\partial x \\ \partial u/\partial y + \partial v/\partial x & 2\partial v/\partial y & \partial v/\partial z + \partial w/\partial y \\ \partial u/\partial z + \partial w/\partial x & \partial v/\partial z + \partial w/\partial y & 2\partial w/\partial z \end{bmatrix} \quad (2)$$

Let us now consider a small material surface of area A within the flowfield that is arbitrarily oriented with respect to the local velocity vector. The orientation of this surface can be defined in terms of three, mutually perpendicular, unit vectors $\vec{n}_1, \vec{n}_2, \vec{n}_3$--$\vec{n}_1$ and $\vec{n}_2$ are oriented parallel to the surface will $\vec{n}_3$ is normal to the surface. The components of the shear stress tensor will induce a shearing force, $\vec{F}_S$, on this surface which can be expressed in terms of its components along each of the three unit vectors ($\vec{F}_s = \vec{F}_{s_1} + \vec{F}_{s_2} + \vec{F}_{s_3}$):

$$\vec{F}_{s_k} \equiv \bar{\bar{\tau}} \cdot \vec{n}_k = \begin{bmatrix} \tau_{xx}n_{k_x} + \tau_{xy}n_{k_y} + \tau_{xz}n_{k_z} \\ \tau_{xy}n_{k_x} + \tau_{yy}n_{k_y} + \tau_{yz}n_{k_z} \\ \tau_{xz}n_{k_x} + \tau_{yz}n_{k_y} + \tau_{zz}n_{k_z} \end{bmatrix} \text{ for k = 1, 2, 3} \quad (3)$$

where n_{k_x}, n_{k_y}, and n_{k_z} are the three Cartesian components of the $\vec{n}_k$ unit vector. The above serve to demonstrate that in a complex three-dimensional flow, the shearing force acting on a arbitrarily oriented surface has components along all three spatial directions. Furthermore, it is important to emphasize that in order to determine the total flow-induced force on the material surface under consideration we should also consider the contribution of the pressure field which creates a force acting along the $\vec{n}_3$ direction, i.e. the direction normal to the surface.

Let us consider now a fish travelling with velocity $\vec{V}_f$ within a complex three-dimensional flow environment. With respect to an observer that moves with the fish the ambient flow velocity is $\vec{V} - \vec{V}_f$. Thus, eqns. (1) and (2) can be readily applied to obtain the velocity gradient and stress tensors as observed by the fish simply by replacing u, v, and w with u-u_f, v-v_f, and w-w_f. If the material surface we considered in our previous discussion is now thought to represent a small panel on the fish skin, eqn. (3) can be used to calculate the three components of the shearing forces acting on that panel. Since the fish bodies exhibit both longitudinal and transverse curvatures, the orientation of the $\vec{n}_1, \vec{n}_2, \vec{n}_3$ vectors changes continuously as we move along the fish skin. For that reason it is more convenient to express the resulting shearing forces in terms of a coordinate system whose origin is attached at a fixed point on the fish body (say the center of gravity of the fish) and its axes are always oriented along the principal and two minor axes of the fish body (see fig. 2). Clearly the total shearing force will have, in the general case, three non-zero components along each one of these axes. The component along the principal axis of the fish (from head to tail) will act to stretch or compress the fish body, leading to possible decapitation and descaling, and could also induce bending loads. The components along the two lateral directions, on the other hand, would tend to compress or stretch the fish laterally as well as produce twisting and/or bending loads. It is, therefore, clear from these general qualitative arguments that meaningful shear experiments, that are representative of real-life situations encountered by passing fish in powerplants, should be able to reproduce, quantify, and correlate with specific injuries all these possible three-dimensional loads.

Most previous shear experiments have been conducted by introducing fish in the turbulent shear layer generated by a high-speed jet discharging into a tank of stagnant water (Groves, 1972; Turnpenny et al., 1992)--experiments involving the flow between two co-rotating cylinders have also been reported (Morgan et al., 1976), but these have focused on the effects of mild shear on eggs and larvae of bass and perch. In such a flowfield the predominant components of the mean (time averaged) velocity gradient and shear-stress tensors are *du/dy* and τ_{xy}, respectively. It is important to emphasize, however, that in the vicinity of the jet the instantaneous flowfield, which is the actual environment encountered by the fish, is very complex, highly three-dimensional, unsteady, and characterized by a broad range of spatial and temporal scales. In other words, in such experiments, and for that matter in any flowfield in which strong mean velocity gradients exist, it is very difficult, if not impossible, to isolate the effects of "shear" from those of "turbulence"--this point is further discussed in more detail in the subsequent section.

The main limitation of such experiments is the lack of any flow measurements. As a result, the jet velocity is the only parameter that has been traditionally used, (Groves, 1972; Turnpenny et al., 1992) to estimate the magnitude of the shearing force experienced by the fish and develop correlations with observed adverse effects, such as dissorientation, severe injuries and even mortality. Although this velocity can be used to come up with an order of magnitude estimate of the mean shearing force experienced by the fish, it is important to recognize that any particular injury could have been the result of an instantaneous extreme turbulent fluctuation which could very well be considerably higher than the mean. To estimate the likelihood and intensity of such events, therefore, it is necessary to supplement whatever method of fish tracking and observation is employed with detailed mean velocity and turbulence statistics measurements.

Moreover, although jet experiments can expose fish into very high levels of shearing forces and turbulence fluctuations, they do not necessarily represent, neither quantitatively nor qualitatively, all real-life situations encountered within a powerplant. Quantitatively similar flowfields, in the sense that they exhibit a single dominant component of the mean shear-stress tensor, could of course occur in regions within the powerplant such as near gaps at the downstream end of overhanging wicket gates or hub gaps of Kaplan turbines, where the local geometry could possibly induce high-speed jet-like flowfields. Clearly, however, these experiments are not representative of situations such as those a fish could encounter within a swirling flow core entering a region of strong adverse pressure gradients (draft-tube inlet), or within strong longitudinal vortices developing along a wall, like those emanating from tip clearance gaps, etc.. Such regions of the flow are highly three-dimensional, with most components of the mean velocity gradient tensor being comparable to each other, with distinctly different, as compared to jet configurations, turbulence structures. For that reason it is important that any attempt to design controlled laboratory experiments with real fish begins with a detailed examination of the flow within the various components of the powerplant in order to identify and carefully classify, based on their overall structure, all local flow scenaria a fish is likely to encounter during its passage. This zonal approach will naturally lead to a number of different laboratory experiments with real fish, each designed to reproduce the features of the various flow zones within the powerplant and study the type and intensity of loads they impart on passing fish.

3.4 Turbulence induced loads

Turbulence induced fish damage is probably the least understood and most commonly misunderstood injury mechanism. Few simple experiments conducted by introducing "turbulence" into a chamber via a series of water jets (Killgore et al, 1987) have shown that although there is practically no dependence of fish mortality on the frequency of the turbulent fluctuations, there is significant dependence on "turbulence" intensity. To facilitate a more in-depth critical look in such experiments, let us briefly start by discussing some fundamental aspects of turbulent flows.

Turbulent flows are unsteady, three-dimensional, and characterized by a broad range of spatial and temporal scales. Within a powerplant, the largest turbulent scales are associated with eddies whose size is comparable to a characteristic dimension of a given subsystem (say, the height of the runner blades or the diameter of the draft-tube, etc.). The smallest scales (the so-called Kolmogorov scales), on the other hand, are those eddies whose energy is dissipated directly into heat by the molecular viscosity of the fluid. The resulting disparity in scales along with the complex, highly non-linear transfer of energy among them is what makes turbulent flows extremely difficult to study and understand. In powerplants this disparity is further exaggerated by the large size of real-life hudroturbines and the very high Reynolds numbers--the smallest turbulent scales tend to become smaller as the Reynolds number increases and the relative importance of the molecular viscosity of the fluid diminishes.

The most common engineering approach for describing a turbulent flowfield is the so-called Reynolds decomposition. The instantaneous values of the various flow quantities (velocity components and pressure) are expressed as a sum of a mean value plus a fluctuating, about the mean, component of variable frequency and intensity. It is important to recognize that in hydroturbines the so resulting mean value is not necessarily constant in time but it contains large scale organized temporal changes in the flow, such as those induced by the rotation of the runner. By substituting the Reynolds decomposition of the velocity components and pressure into the instantaneous equations of motion (continuity and momentum) and averaging the resulting equations, we obtain the so-called Reynolds-averaged Navier-Stokes (RANS) equations. The resulting equations look very similar to the original instantaneous Navier-Stokes equations but they are formulated in terms of the mean velocity and pressure components (eqn. (1) and (2) are still applicable but now they represent the mean velocity gradient and shear-stress tensors, respectively). Most importantly, the Reynolds-averaged momentum equations contain additional terms, the so-called Reynolds stresses, that represent the effects of turbulent velocity fluctuations on the evolution of the mean flow quantities. We refer to these terms as stresses because like the mean shear-stress tensor given by eqn. (2), which represents transport of momentum by molecular action in the flow, they too represent momentum transport but due to the action of turbulent eddies. Since turbulence is far more effective in transporting momentum and enhancing mixing, the Reynolds stresses are, for the most part of the flowfield, considerably larger than the viscous stresses. The production and evolution of the Reynolds stresses are governed by a set of very complex transport equations, which contain the mean velocity components and their gradients, and turbulent correlations involving Reynolds-averaged products of velocity and pressure fluctuations as well as triple fluctuating velocity products. The importance of these equations for our discussion lies in the fact that they clearly demonstrate that Reynolds-stresses, and, thus, turbulence, can be produced and sustained only when gradients in the mean velocity exist.

The above discussion has important implications insofar as fish experiments are concerned. First, it is obvious that it is very difficult to design experiments in which the effects of mean shear as an injury inducing mechanism can be separated from those of turbulence, as the former is responsible for producing and affecting the structure of the latter. Although in principal experiments could be carried out by introducing fish into a homogeneous turbulence field (which in the absence of mean shear will decay in time), it is unlikely that such a situation will ever occur in a real powerplant where the geometrical complexities induce intense velocity gradients and, thus, continuously produce turbulence. Therefore, rather than trying to isolate the effects of mean shear from those of turbulence we, once again, recommend that the zonal experimental approach described in the previous section is adopted. By carefully designing controlled laboratory experiments that reproduce various local flow scenaria within the powerplant, we can ensure that fish will be exposed to flow environments whose mean flow and turbulence structures are broadly similar to typical real-life situations.

It should be recognized, however, that meaningful quantification of turbulence-induced loads on fish bodies in such experiments is a particularly challenging task. This is because out of the broad range of turbulent eddies present within the powerplant, those whose size is comparable to the characteristic length and time scales associated with a given fish will mainly determine the severity of the local loads imparted on the fish body and the extent of possible diss-orientation effects. The typical length and time scales of such "*fish-sized*" eddies are determined by the size of the fish and the relative, with respect to the fish, ambient flow velocity. Turbulent eddies smaller than a "fish-sized" eddy are, from the fish standpoint, small-scale noise. Much larger eddies, on the other hand, contribute in the general motion of the fish within the flowfield, while intermediate size eddies could cause mild diss-orientation. Therefore, quantifying the potential for turbulence-induced injuries and diss-orientation requires information about the so-called spectral content of turbulence, that is the manner in which turbulent energy is distributed among the various eddies in the flow. Generally speaking, the more energetic "fish-sized" eddies are the more likely it is that they could inflict serious damage on passing fish.

3.5 Mechanical injuries

The term mechanical injuries refers to all injury events caused when a fish comes in contact with a solid surface. These include direct strike with a turbine blade, impact and scrape on walls, piers etc. Such events can induce injuries, mortality, or be completely harmless, depending on the angle the fish strikes the solid surface and its velocity at the time of impact. Several types of injuries are observed in the case of contact with a wall or blade, ranging from descaling (result of scraping against a rough surface like a cement wall, or grinding, i.e. passage through a narrow opening like a blade-hub gap), to decapitation and fatal spinal injuries.

Laboratory experiments performed to investigate mechanical injuries (Turnpenny et al. 1992), have shown that mortality is proportional with the velocity of impact, and inversely proportional with the thickness of the blades. Moreover, the inertia of the fish was found to directly determine the likelihood of avoiding wall contact, since lighter fish can follow the flow streamlines much easier than heavier fish. For instance, Turnpenny et al (1992) reported that, for identical geometrical configuration and flow conditions, fish heavier than 0.2 kg exhibited a 75 percent strike probability, whereas for lighter fish ($\approx$0.02 kg) this number was reduced to approximately 13 percent.

Field experiments are significantly more difficult to conduct, because of the geometrical and flow complexities encountered in an actual powerplant. We can classify such experiments in two broad categories based on whether the powerplant is treated as a "black box," or whether attempts are made to track the exact passage of fish inside the various subsystems. In the former approach, fish are introduced at specified upstream locations near the intakes and are subsequently collected at the tailrace. The fish are tagged by balloons or other means in order to facilitate the retrieval. This method of experimentation has provided valuable results in the forms of identification of realistic types of injuries suffered and overall mortality rates, (Mathur et al., 1996; Heisey et al., 1995, 1996; Schoeneman et al., 1991). However, it is obvious that such a "black box" type of approach provides little or no information of what actually happens inside the powerplant.

The latter approach attempts to elucidate this very problem, that is to determine the actual fish trajectories through the entire powerplant and record all events that take place along them. Obviously such an approach poses considerable challenges to experimentalists: the environment in the plant is absolutely dark, very turbulent, often cavitating, the fluid is moving very fast and exerts large forces on any recording device introduced in the flowfield. Original attempts to use high-speed photographic equipment, (Moore and Scott, 1988; Vaughn, 1995), were inconclusive due to the "unnatural" effect the necessary lighting (dim as it might be with the new low-lux video equipment) has on fish behavior. Moreover, the actual environment in the powerplant does not allow for photography of objects at distances greater that 2-4 feet from the lenses. Suggestions to use low intensity, fish-attached illuminating devices for fish tracking (like LED elements, etc.) may improve the capability for tracking the trajectory of fish and for impact events recording. Even this technique, however, will not provide information injury mechanisms other than mechanical strike and scrape, as it does not provide any information regarding the flow environment encountered by the fish. A second technique that has been used to study fish passage is through the employment of hydroacoustic equipment. Based on the SONAR concept, hydroacoustics have been used extensively in open sea fishing vessels to detect fish schools with success. The use of such equipment for hydropower plant fish passage detection has the great advantage of being non-intrusive and, in case of split-beam acoustics, of being capable to track the full three-dimensional motion and velocity of the fish in real time, (Ransom and Steig, 1994). The correct application of this technique is far form being straightforward however: cavitation, background noise, fish species confusion and resolution capability (as far as small fish species are concerned) are some of the problems that make the application of this technique very challenging. Advanced pattern recognition software, expensive equipment and highly skilled personnel (both at the installation-operation level and at the results interpretation level) are necessary for the successful application of this technique. Despite these difficulties, it is however the most promising *in situ* fish tracking method reported so far.

Finally, we should point out that very few attempts have been made to tackle the problem via theoretical or numerical approaches. Most theoretical efforts have concentrated on regression analysis of experimental data (Turnpenny et al., 1992; USACE, 1991) and subsequent formulation of laws and rules that are supposed to govern fish passage. More recently efforts have been made to correlate the fish dimensions with the geometrical characteristics of a blade-to-blade passage by taking into account possible effects of fish orientation, (Fisher et al., 1997).

The above literature review underscores the enormous complexity of the problem and the numerous challenges that need to be overcome for conducting meaningful laboratory and field experiments. In that regard, the subsequently described numerical model provides, for the first time, a comprehensive, albeit simplified in some instances, framework that could facilitate the systematic assessment of most possible injury mechanisms. It should be made clear, however, that the proposed numerical model requires considerable input from carefully designed controlled laboratory experiments for calibration and validation purposes before it can be employed as a reliable engineering tool. Such experiments are identified in a subsequent section at the end of this report.

4. Description of the method

The numerical model requires as input a complete three-dimensional CFD solution of the flow through the entire powerplant at a given operating point. The computed flowfield is described in terms of pressure, mean velocity components, and turbulence statistics. These flow quantities are defined on an assembly of computational blocks, each corresponding to a powerplant subsystem (e.g. forebay, intake, stay vanes, wicket gates, runner, draft-tube, and tailrace). Given the geometrical complexity of a typical powerplant, it is very common that a large number of sub-blocks are necessary to adequately describe each one of these subsystems. For that reason the model has been designed so that it can accommodate an arbitrary number of blocks and block sizes, limited only by the available computer memory.

Within this pre-computed virtual flow environment, fish are introduced, one after the other, at user-specified locations, initial velocities and orientations. The trajectory of each fish is computed using a fully three-dimensional tracking algorithm. The motion of each fish is described in terms of a sequence of translations and rotations along the three Cartesian axes, and, thus, a total of six differential equations (for the Cartesian components of the linear and angular acceleration vectors) are necessary for describing the entire spectrum of possible motions. The source terms in these equations represent the various forces exerted by the flow on the individual fish. At every point along the computed trajectory the local flow conditions experienced by the fish are recorded. The application of this algorithm continues until one of the following events occurs:

1) the fish exits the computational domain,
2) the fish impacts a solid wall or blade with a velocity and angle of attack that exceeds a pre-specified threshold and is assumed dead.

It is evident from the above brief description that the algorithm consists of several modules that need to be carefully formulated for realistic simulations. These include the: i) selection of a plausible fish geometry representation ii) formulation and accurate and efficient numerical solution of the equations of motion and iii) implementation of a physically meaningful model that accounts for wall impact and scrape events (collectively denoted herein as the "bounce-back" model). The modeling strategies adopted for each of these modules are described in detail in the subsequent sections.

4.1 "Virtual Fish" geometry and characteristics

Real life fish are bodies of extreme complexity, with a non-analytical shape, a plethora of appendages (fins, gills etc.), openings and a complex surface roughness distribution due to the scales. It would be very difficult to account for all these complexities in an engineering design tool that needs to be fast and efficient. For that reason, we have chosen to approximate the actual fish body with a prolate ellipsoid, shown in figure 1, which, in our subsequent discussion, will be referred to as the *Virtual Fish* (VF). The VF retains all of the main features of the actual fish: its length, width and height, its weight (and weight distribution), its buoyancy characteristics and, at least for part of the model, its surface roughness.

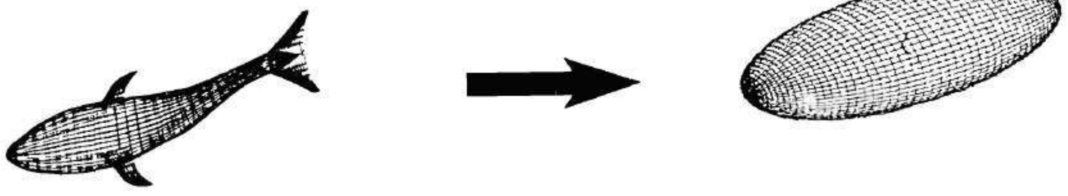


Figure 1. Transformation from real to virtual fish

The equation that describes the VF geometry is of the following form:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1 \tag{4}$$

where a, b and c correspond to the three principal axes of the body.

Some geometrical characteristics of interest, that are employed in the subsequently described model, include the VF volume, V, and its three moments of inertia--I_x, I_y, and I_z, where x, y, and are the VF principal axes, see figure 2. For a body described by eqn. (4), these quantities are given as follows:

$$\begin{aligned} V &= \frac{4}{3}\pi abc, \\ I_x &= \frac{1}{5}V(b^2+c^2), \\ I_y &= \frac{1}{5}V(a^2+c^2), \\ I_z &= \frac{1}{5}V(a^2+b^2), \end{aligned} \tag{5}$$

The VF surface is discretized using a structured two-dimensional surface mesh as shown in figure 1. It is important to point out that each node of this mesh can be link to a specific fish body part. Assuming, for example, that the body of a 15 centimeter juvenile salmon is discretized using a 6x6 (along the head-to-tail and girthwise directions, respectively) mesh, node (2,1) can be used to mark the left eye of the fish, nodes (2,3) and (2,4) define the right gill, while the midbody spine is located between nodes 3,2 to node 5,2. Such a topological mapping, which can be specified as desired by the user, allows the interpretation of the results in terms of biological rather than geometrical terms and facilitates the classification of specific types of injuries.

4.2 Formulation of the equations of motion

Since the VF is a three-dimensional object, six differential equations are necessary for the full description of its motion: three linear acceleration components (for the translatory motion along the three Cartesian axis), along with three angular acceleration components (for the rotational motion around the aforementioned axis). Assuming steady flow, these equations are formulated as follows:

$$
\begin{aligned}
& m\frac{du_i}{dt} = F_D^i + F_L^i + F_{SH}^i + F_P^i + F_{AM}^i + F_B^i + \cdots,\ i = 1,2,3 \qquad \textit{and} \\
& I_{\hat{x}}\frac{d\omega_{\hat{x}}}{dt} - \omega_{\hat{y}}\omega_{\hat{z}}(I_{\hat{y}} - I_{\hat{z}}) = T^h_{\hat{x}}\ , \\
& I_{\hat{y}}\frac{d\omega_{\hat{y}}}{dt} - \omega_{\hat{z}}\omega_{\hat{x}}(I_{\hat{z}} - I_{\hat{x}}) = T^h_{\hat{y}} \qquad \textit{and} \\
& I_{\hat{z}}\frac{d\omega_{\hat{z}}}{dt} - \omega_{\hat{x}}\omega_{\hat{y}}(I_{\hat{x}} - I_{\hat{y}}) = T^h_{\hat{z}}
\end{aligned}
\tag{6}
$$

where: m_b is the mass of the fish; *d/dt* is the Lagrangian derivative; u_i are the Cartesian components of the fish velocity vector; $\hat{x}, \hat{y}, \hat{z}$ are the coordinates of a system that coincides with the principal axes of the fish (see figure 2); $\omega_x, \omega_y, \text{and}\, \omega_z$ are the angular velocities around each principal axis; and I_x, I_y, and I_z are the principal moments of inertia defined by eqns. (5). The terms in the right hand side of eqn. (6) represent the various forces and torques acting on the VF at a given point along its trajectory. These include, forces due to: i) viscous drag, F_D^i, and lift, F_L^i ; ii) ambient mean shear in the flow, F_{SH}^i ; iii) ambient pressure gradient in the flow, F_P^i; iv) added mass effects, F_{AM}^i; and v) buoyancy, F_B^i. The dots in eqn. (6) represent higher order forces that are typically difficult and time consuming to compute while their overall effect on the fish trajectory is fairly small. For the sake of expedience and computational efficiency such forces are neglected herein.

Assuming that the various forces acting on the VF are known, eqn. (6) can be integrated in time to obtain the VF velocities at the new time step. The new position of the VF can subsequently determined by integrating in time the following equations for the Cartesian components, x_{bi}, of the VF position vector:

$$\frac{dx_{bi}}{dt}=u_{bi}\,,\ i=1,2,3 \tag{7}$$

The equations governing the rotational motion of the body are formulated in terms of the body fitted principal-axes coordinate system $(\hat{x},\hat{y},\hat{z})$ and thus their integration requires special treatment. For this purpose, we introduce a co-motion coordinate system, whose origin coincides with the origin of the principal-axes coordinate system, located at the VF rotation center, and its axes, x', y', z', remain always parallel to the original Cartesian system (see figure 2)--that is, the co-motion coordinate system translates with the VF velocity but does not rotate. Then the transformation between the co-motion system and the fish rotational system is given by $\hat{x} = Ax'$, where A is the transformation matrix defined as follows:

$$A=\begin{bmatrix} 1-2(\varepsilon_2^2+\varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2+\varepsilon_3\eta) & 2(\varepsilon_1\varepsilon_3+\varepsilon_2\eta) \\ 2(\varepsilon_2\varepsilon_1+\varepsilon_3\eta) & 1-2(\varepsilon_3^2+\varepsilon_1^2) & 2(\varepsilon_2\varepsilon_3+\varepsilon_1\eta) \\ 2(\varepsilon_3\varepsilon_1+\varepsilon_2\eta) & 2(\varepsilon_3\varepsilon_2+\varepsilon_1\eta) & 1-2(\varepsilon_1^2+\varepsilon_2^2) \end{bmatrix} \tag{8}$$

In the above equation, $(\varepsilon_1, \varepsilon_2, \varepsilon_3, \eta)$ are the so-called Euler's quartenions defined as:

$$\begin{aligned} [\varepsilon_1,\varepsilon_2,\varepsilon_3]^T &= \bar{e}\sin\left(\frac{\phi}{2}\right) \\ \eta &= \cos\left(\frac{\phi}{2}\right) \end{aligned} \tag{9}$$

where $\vec{e}$ is the unit vector along the axis of rotation, and ϕ is the angle of rotation. Since there are only three degrees of rotational freedom, Euler's quartenions must satisfy the following constraint:

$$\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \eta^2 = 1 \tag{10}$$

By the definition of the co-rotation system and due to the fact that the VF rotates continuously as it is transported through the flow, the Euler's quartenions change in time at a rate that is given by the following set of ordinary differential equations:

$$\begin{bmatrix} \dot{\varepsilon}_1 \\ \dot{\varepsilon}_2 \\ \dot{\varepsilon}_3 \\ \dot{\eta} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \eta\hat{\omega}_x - \varepsilon_3\hat{\omega}_y + \varepsilon_2\hat{\omega}_z \\ \varepsilon_3\hat{\omega}_x + \eta\hat{\omega}_y - \varepsilon_1\hat{\omega}_z \\ -\varepsilon_2\hat{\omega}_x + \varepsilon_1\hat{\omega}_y + \eta\hat{\omega}_z \\ -\varepsilon_1\hat{\omega}_x - \varepsilon_2\hat{\omega}_y - \varepsilon_3\hat{\omega}_z \end{bmatrix} \tag{11}$$

Finally, the initial conditions necessary for the rotations integration are obtained by the following relations, assuming that $\eta \neq 0$:

$$\begin{aligned} \eta &= \pm\frac{1}{2}(1 + a_{11} + a_{22} + a_{33})^{1/2} \\ \vec{\varepsilon} &= \frac{1}{4\eta}\begin{bmatrix} a_{23} - a_{32} \\ a_{31} - a_{13} \\ a_{12} - a_{21} \end{bmatrix} \end{aligned} \tag{12}$$

In the above equations (12), a_{ij} correspond to the elements of the transformation matrix A of equation (8).

The details of the numerical technique employed to integrate eqns. (6) and (7) are given in section 4.4 below. The formulation of the equations used to calculate the various forces in the right hand side of eqn. (6) is described in the next sections.

Once the fish species under consideration has been properly described and all its properties initialized, the initial positions of all nodes are located in the CFD grid topology using the given initial position and orientation of the fish along with eqn (12). Then all necessary quantities are interpolated, forces and torques are computed and eqns (6) and (7) are integrated to the next time step. Eqns (8) to (11) are then used to estimate the exact position for all surface node of the fish and the procedure is repeated for the next time step.

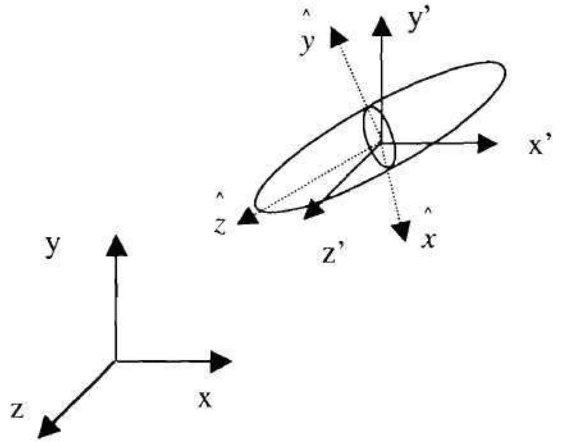


Figure 2. Coordinate systems

4.3 Description of driving forces

The various forces acting on the Virtual Fish that drive its passage through the powerplant and correspond to the source terms of the equations of motion are presented in the sequel. The forces described and used in the model correspond to the most important contributions present. Basset- (unsteady memory effects), Magnus- (rotation-induced lift) and turbulence-induced forces are omitted in the present version of the model. It is our intention, however, to carefully investigate the significance of these forces in the second phase of this work and take into account those that are found to be significant. Given the modular structure of the model, additional forces can be readily implemented in the right hand side of the equations of motion.

4.3.1 Force due to viscous flow around the fish

This force accounts for the effects of the three-dimensional flowfield that develops locally around the fish body and consists of contributions from both shearing and pressure forces. Since the fish-induced flowfield is not known in the present model, we estimate this force by assuming that it is equal to the force that would be exerted on the fish body if a uniform ambient flow was approaching the fish at an incidence angle equal to the local incidence angle at the current location of the fish within the powerplant. The so resulting force is expressed as the sum of a lift, $\vec{F}_{Lf}$, and a drag, $\vec{F}_{Df}$, force acting along the direction of and perpendicular to the relative velocity vector ($=\vec{V}-\vec{V}_{VF}$, where $\vec{V}$ and $\vec{V}_{VF}$ are the fluid and VF velocity vectors), respectively, that is:

$$\vec{F}_{ff} = \vec{F}_{Lf} + \vec{F}_{Df} \tag{13}$$

The magnitude of the lift and drag forces is determined as follows:

$$\begin{aligned} \vec{F}_{Lf} &= \frac{1}{2} C_{Lf} \rho S_{VF} \left|\vec{V} - \vec{V}_{VF}\right|^2, \\ \vec{F}_{Df} &= \frac{1}{2} C_{Df} \rho S_{VF} \left|\vec{V} - \vec{V}_{VF}\right|^2 \end{aligned} \tag{14}$$

where S_{VF} is the VF frontal area, ρ is the fluid density, and $C_{L,D}$ are the VF lift and drag coefficients, respectively.

Accurate determination of these forces is a particularly challenging problem as it requires knowledge of the lift and drag coefficients of the VF body under arbitrary surrounding flow conditions. Since these coefficients depend on the VF geometry, its local Reynolds number (defined as $\mathrm{Re}_{VF} = \left\| \vec{V} - \vec{V}_{VF} \right\| L_{VF} / \nu$) and the flow angle of incidence, we need to device correlations that provide us with their values for every possible angle of attack and Reynolds number for every possible fish geometry. Such a correlation can be derived via either laboratory experiments or detailed three-dimensional Navier-Stokes computations of the flow around a VF body over the entire range of possible free-stream conditions. Neither approach, however, is economical as it is estimated that in order to cover just the angle of attack parameter, for a fixed Reynolds number and a given VF geometry, a set of at least 6x6=36 measurements or computations will be required to obtain a moderate resolution of 15° rotation increments (6 measurement points for rotation around each one of the transverse principal axes of the VF). For that reason, we chose to construct approximate correlations by compiling available experimental measurements for prolate spheroid (VF-like) bodies and real fish.

Based on experiments with real fish, fish biologists have proposed similar correlations that link the flow-induced force with the fish geometry at zero angle of attack--see (Childress, 1977; Wu et al., 1974; Hoerner, 1965), among others. Such relations are of the following general form:

$$C_{D_o} = C_f(1. + 1.5(l/d)^{(3/2)} + 7(d/l)^3) \tag{15}$$

where C_{D_o} is the drag coefficient at zero incidence, *l/d* the fish body length-to-diameter ratio and C_f is the friction coefficient for a flat plate at the same Reynolds number.

There is a significant amount of experimental and computational data on the lift and drag forces acting on prolate spheroids for angles of attack ranging from 0° to almost 90°, with more detailed data for the regime up to 30°, (Su et al., 1993; Costis et al., 1989; Fu et al., 1994; Meier and Kreplin, 1980) and others. Using regression analysis, most available data in the literature can be fitted, with relatively small dispersion, by simple quadratic equations linking the forces exerted on the body by the fluid with the angle of attack:

$$\begin{aligned} C_{L_\theta} &= \alpha_L\,\theta^2 + \beta_L\theta \\ C_{D_\theta} &= \alpha_D\,\theta^2 + \beta_D\theta + 1 \end{aligned} \tag{16}$$

where C_{L_θ} and C_{D_θ} are corrections for the lift and drag coefficients due to non-zero angle of attack, θ, and $\alpha_{L,D}$, and $\beta_{L,D}$ are regression parameters, currently with values equal to – α_D=-0.483, β_D=0.514, α_L=-0.091, and β_L=0.443, respectively. It is important to point out that due to lack of suitable experiments, the above correlations do not account for the effects of: i) laminar-to-turbulent flow transition, possibly occurring somewhere along the fish body; and ii) unsteady flow, which for prolate spheroid geometries has been observed in experiments to occur for $\theta>60^\circ$.

The final lift and drag coefficients used in eqns. (12) are calculated as follows:

$$C_{Lf} = C_{D_o} C_{L_\theta},\ \ C_{Df} = C_{D_o} C_{D_\theta} \tag{17}$$

Although at first glance this methodology appears somewhat crude, it is, given the lack of more detailed measurements, the only reasonable and computationally efficient approach to tackle an otherwise unsolvable problem. As more data become available, either from experiments (see section 7 below) or from in-house computations, they can be very easily incorporated in the model and enhance the accuracy of the aforementioned regression formulas.

4.3.2 Force due to ambient pressure gradient

In a complex three-dimensional flow environment, the pressure sensed by the various sides of the VF is not uniform but depends on the local pressure gradients in the flow. By integrating the pressure on the fish surface we obtain a force due to the ambient pressure gradient that contributes to its motion. This is done by simply multiplying the area of each fish surface panel with the average of the four vertex pressures that define that panel and by the normal unit vector of that panel, resulting in three force components, which are summed for every surface fish panel. It should be noted that because of the way most CFD codes treat pressure, this term does not include the effect of buoyancy, and should be viewed as the integral of the difference of static pressure minus hydrostatic pressure.

4.3.3 Force due to added mass effect

In order to account for the response of the fluid surrounding the VF to acceleration, the so-called added mass effect, an additional force term is introduced as follows:

$$\vec{F}_{AM} = a\forall_{VF}\rho_{water}\frac{d}{dt}\left(\vec{V} - \vec{V}_{VF}\right) \tag{18}$$

where $\forall_{VF}$ is the volume of the fish and a is the added mass coefficient, (Newman, 1977). The velocity derivative along the three directions is computed via a first order accurate finite difference scheme, from current and stored fish velocity values.

The numerical integration of the equations of motion (eqn. (6)), can be greatly simplified and stabilized by moving the added mass force to the left hand side of the these equations. This amounts to substituting the mass of the fish with a new effective mass that accounts for both the inertial mass and the added mass.

4.3.4 Buoyancy force

The net buoyancy force (effective fish weight) acts in the vertical direction and is computed as follows:

$$\vec{F}_B = V_{fish}\left(\rho_{fish} - \rho_{water}\right)\vec{g} \tag{19}$$

where $\vec{g}$ is the gravitational acceleration. In the coordinate system used in our CFD simulations, $\vec{g} = (g,0,0)$. The density of the VF is computed from its dimensions and weight. We should note that the modular structure of the method makes it very easy to take into account situations of variable buoyancy, which is the case when the fish changes (willingly or unwillingly) the air content of its bladder. Lack of knowledge of the biological laws that dictate such changes made us omit this capability from the current version of the code.

After composing all the forces acting on the fish, we are left with three Cartesian components of the sum force and their point of action. With this set, the system of equations presented in (6) and (7) is closed and can be integrated appropriately.

4.4 Numerical integration of the equations of motion

The governing equations of a motion, eqns. (6) and (7), are integrated in time in a Lagrangian fashion. That is, unlike the governing flow equations which are solved on a fixed Eulerian mesh, the solution of eqns. (6) requires the calculation at every time step of the fish position, velocities etc. This implies that any numerical scheme to be used for this purpose should consist of two components: i) a temporal integration scheme for advancing in time eqns. (6) and (7) and to achieve that it is necessary to employ ii) an algorithm for searching and interpolating in space. As is the case with all our modeling choices in this work, the selection of an appropriate numerical scheme was guided by the need to balance computational efficiency and numerical accuracy.

4.4.1 Temporal integration scheme

Extensive numerical experiments with temporal integration schemes showed that schemes that are second order accurate and higher yield identical results for the fish trajectories, provided that the time step is kept sufficiently small. However, schemes whose accuracy is higher that second order require either excessive memory (Euler type schemes) or significantly more computational time (Runge-Kutta, predictor-corrector and other multi-stage type schemes). Both of the above requirements can substantially increase the overall computational overhead, particularly when such schemes are employed to integrate in time the trajectories of many fish, one after the other. Since we found no significant accuracy improvements with the use of a higher-than-second order approximation, the three-point, second-order accurate Euler explicit scheme was selected for integrating both eqns. (6) and (7):

$$\left(\frac{du_{bi}}{dt}\right)^{n+1} \approx \frac{3u_{bi}^{n+1} - 4u_{bi}^{n} + u_{bi}^{n-1}}{2\Delta t} \tag{20}$$

where n denotes the time level and Δt is time step. The time step in eqn. (20) is selected in a manner that guarantees numerical accuracy and stability while minimizes the computational resources required for carrying out spatial searches and interpolations. A module has been introduced in the code that pre-estimates[1] mean fish node traveling times along the three directions of every cell of the CFD computational grid. Consequently, the smaller of these traveling times is chosen as the time step (usually multiplied by a factor of 0.1-0.5, to increase accuracy and take into account inertia effects). This approach yields a very conservative time step estimate but has two major advantages: i) the time step is kept small enough for the temporal integrator to be accurate and stable; and ii) it guarantees that the spatial position of a given fish at the new time level will be in the close neighborhood of its current position.

A small example can illustrate clearly the speedup achieved by selecting the time step as described above. Assuming that the new position of a particular node on the fish surface will be within, say, 4 computational cells from the old one, the required search area consists of $(4+1+4)^3$=729 cells (four cells upstream, the current cell and four cells down stream, for all three spatial directions). If our estimation involves a neighborhood of 10 computational cells, we get a total of $(10+1+10)^3$ =9,261 cells to be searched. Arbitrarily defined, user-specified time step requires a searching area that spans 10-15 cell neighborhoods in every direction. The time step selected using the above procedure allows the use of just 1 cell neighborhoods, which implies that the total number of grid cells to be scanned is $(1+1+1)^3$=27. Since the search algorithm takes up more that 60% of the total CPU usage of the model, it is obvious

[1]This is done only once, in the beginning of each run

that a speedup of 0.75X(9,261/27)=250 per time step is achieved through this technique. Of course the final speedup of the model is reduced by a factor 10-15 because the smaller time step means increased number of time steps required for the completion of each trajectory. Still, a significant overall speedup of approximately 15 has been achieved though the use of this technique.

It is important to point out that the code is constructed in such a way, that if the initial 1-cell-neighborhood fails, then all of the computational domain is searched. This happens very rarely, however, and usually only for newly injected fish. The actual mechanism that this position finding takes place is described in the next section.

4.4.2 Spatial search and interpolation algorithm

In order to be able to estimate the local flow conditions around every fish surface mesh node, we need to pinpoint the location of that node in the computational flow field. This is not an easy task, since computational grids for draft tubes are in general curvilinear, skewed, stretched and very irregular. The technique used to find the grid cell that the fish is in is based on an equality of volumes principle. Each of the grid cells is subdivided in six tetrahedra that span the original volume. Then, the searching algorithm assumes that the center of the fish is in every one of these tetrahedra and defines four new tetrahedra for each of the initial ones. The four vertices of the new sub-tetrahedra correspond to three vertices of the original tetrahedron and the fish node under investigation. The sum of the volumes of these new four tetrahedra will be equal (within some accuracy depending on roundoff error) to the original tetrahedron volume, if and only if the center of the fish is within this tetrahedron (figure 3). When this is satisfied, we declare the center of the fish to be in that cell and interpolate the values of the variables from the eight grid nodes defining that grid cell. An inverse distance formula, with an exponent of 3.5 is used for the interpolation:

$$\Phi_{node} = \frac{\sum_{nd=1}^{8} \Phi_{nd} d_{nd}^{3.5}}{\sum_{nd=1}^{8} d_{nd}^{3.5}} \qquad (16)$$

where d_{nd} is the distance of the node from every cell vertex.

For very skewed grids, the above search algorithm might fail due to roundoff errors in the calculation of the cell volumes. Our experience so far has shown that this occurs very rarely. In the rare occasion that this happens, the user is provided with a hard-coded constant that can be altered (increased slightly) to accommodate these roundoff errors.

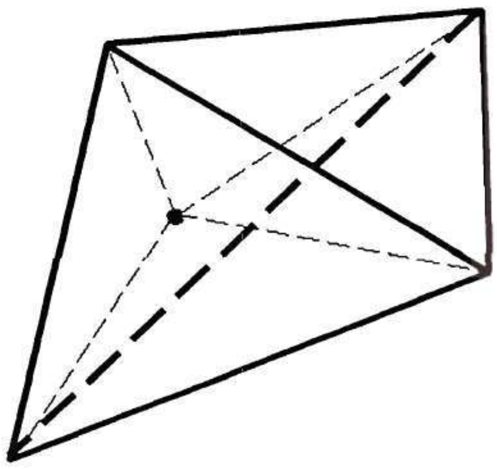


Figure 3. Schematic of the technique used to locate the center of a fish in space

4.5 The bounce back model

The ability of the present approach to account for mechanical strike and scrape events is critical for realistic simulations. For that reason we have incorporated in the model the so-called "bounce-back" module which allows for a VF to impact a solid surface, scrape against it, and continue its motion through the flowfield. The proposed model is conceptually very simple, and, thus, computationally efficient, but it is based on solid kinematic-elastic physical principles. It is described as follows.

Wall impact is assumed when a node of the VF surface grid enters the first layer of grid cells off the solid boundary[2]. The model determines if a node corresponds to a wall node or not

[2] CFD computational grids are very fine near solid surfaces, so this assumption does not restrict the general applicability of the model. On the contrary, it solves very efficiently the problem of determining when a node is "in" or "out" of the computational domain, leading to significant speedup.

by a set of user supplied values, one for each computational cell of the multiblock grid enseble, that are also used by the CFD solver. When an impact event is detected, the VF center of rotation is moved from its original location, CR_b in figure 4, (which is computed using the center of gravity of the fish and the center of application of the various forces on it), to the particular node that impacted the wall, CR_a in figure 4. This means that all subsequent rotations, until the fish leaves the proximity of the wall, will take place around this new point.

The VF motion is computed as usual, with the exception of the introduction of two new forces that account for the presence of the wall:

- a frictional force tangent to the wall (F_{fr} in figure 4).; and
- an elastic 'bounce-back' force perpendicular to the wall, F_{bb} in figure 4.

The magnitude of these forces is calculated as follows:

$$F_{fr} = \eta_f F_{imp} \quad \text{and} \quad F_{bb} = \eta_b F_{imp} \tag{22}$$

where F_{imp} is the force applied by the VF on the wall (see figure 4) and η_f and η_b are empirical coefficient set equal to 0.5 and 0.1 respectively.

During the time interval that VF is in contact with the wall and rotates around one of its surface nodes, it is possible that a second fish surface grid node enters the first near wall grid cells layer. If this occurs, the code sets this second fish surface node as the new center of rotation and computation of motion is continued as described before.

Extensive numerical tests have shown this bounce-back model to behave in a realistic manner. The so computed trajectories are reasonable and broadly consistent with what one would anticipate, based on the conservation of momentum and energy, for the impact of a 3D semi-elastic body on a solid surface. Moreover the model is easy to implement and the constants are very easily adjustable.

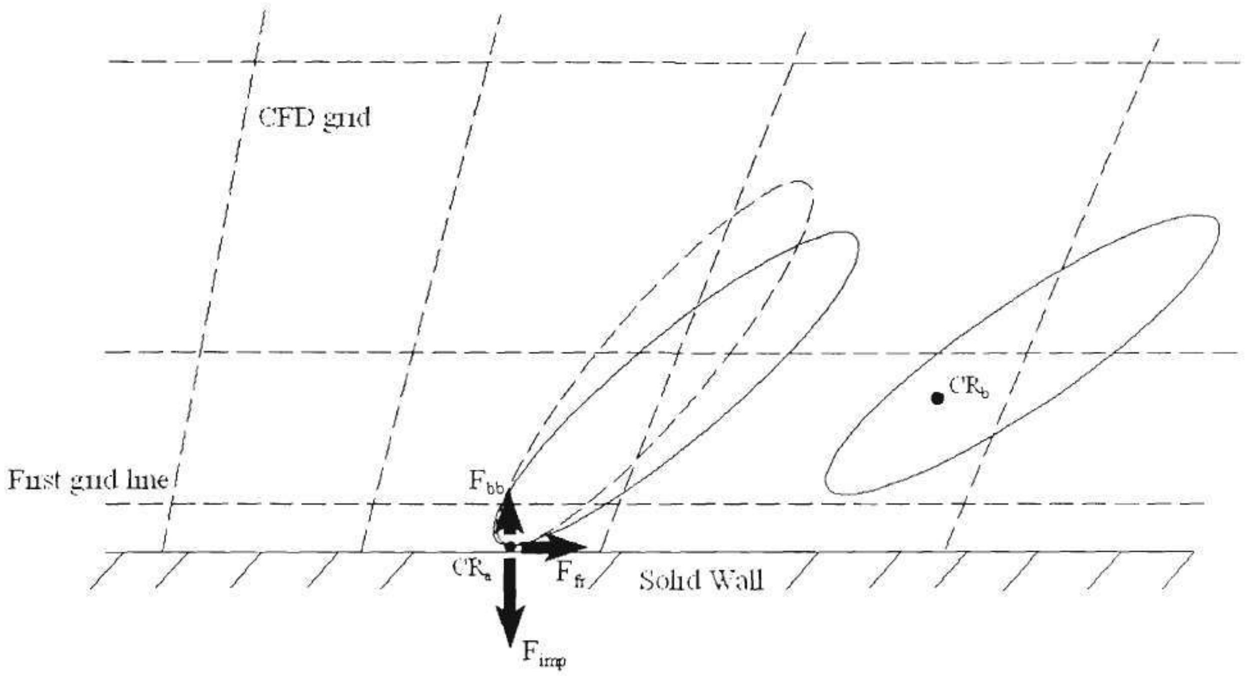


Figure 4. The bounce-back model

5. Pre- and Post-Processing

Each run of the VF model requires as input a set of initial fish release characteristics. These include the location, velocity and orientation at which the fish is introduced in the computational domain and the physical characteristics of each specific fish. Usually, a set of initial locations that forms a 3D surface grid, matching the release locations of interest is generated using a small FORTRAN program. Examples of such initial distributions are provided in Section 6 of the present report. Additionally, the model requires a full 3D solution of the flow within the component to be simulated. Such a solution is produced by a separate CFD algotrithm, comes in terms of velocity components, pressure and turbulent quantities on a set of structured grid nodes and is accompanied by the geometric definition of the aforementioned grid and additional information for the walls that comprise the hydraulic component.

The output of the model consists of an enormous set of time series of various flow quantities and flow-induced loads, stored on each node of the VF body at every position along its trajectory. The coordinates of each node, velocity of the node and of the ambient fluid, pressure, turbulence kinetic energy, a flag indicating impact or scraping along with the forces exerted on the fish from possible impact are the data stored. If we take into consideration the fact that each trajectory consists of several thousands of time steps, the surface of the fish is covered with several dozens of nodes and we might introduce several hundreds of fish for a detailed study, we come up with very large data sets from a single run--approximately 50-100 Mbytes of results are generated for a detailed run. Additionally , for reasons of graphic representation of the solution, an organized set of snapshots of the fish location in the component can be stored. Such time frames can be used for creating animations of the fish trajectory through the machine.

There are several postprocessing techniques that are used with the VF model and allow for an understanding of the characteristics of fish passage. The first thing that the design engineer usually looks for are the actual fish paths and these can be depicted either by plotting the stored x,y,z coordinates of these paths (usually in conjunction with the machine geometry) or by the aforementioned technique of animating a sequence of snapshots of the fish. Commercial software packages like Tecplot or Ensight are used to perform these and all subsequent visualizations. The VF model can produce customized output for one or both of the aforementioned packages simultaneously. The results provided by the post processor, can be classified as follows:

- Trajectory of fish (time history of the location of fish). This set of results corresponds to straightforward recording of the coordinates of one point on or in the fish and when plotted in combination with the CFD mesh topology allows for the determination of the passage of the fish.
- Time histories of the linear and angular velocity of the fish. These series provide information on the velocity and rotation magnitudes experienced by the fish.
- Time histories of the time derivatives of the linear and angular velocity of the fish, which of course correspond to the inertial load the fish experienced during its passage. They serve as an indirect measure of the overall influence the fish experienced during its passage, since, for instance, sudden acceleration or deceleration usually means that a slow-moving fish encounters a rapid-moving water current or vice-versa.
- Cumulative indices of rotations around the three axis. Computed through dividing the total number of times the fish completed a full revolution around an axis by the time necessary for these revolutions, these three numbers are a good measure for fish dizziness.
- Time histories of the flow variables (slip velocity, pressure, turbulence intensity) on every node of the fish surface grid. All these quantities are very useful for the determination of the flow environment the fish (or better yet, every part of the fish body) encountered during its passage.
- Time histories of the maxima or minima of the flow variables (slip velocity, pressure, turbulence intensity) on the fish. By filtering the previous time histories, we can construct a set of time series that instead of being the evolution of a variable on one particular fish node, correspond to the minimum and maximum of this quantity throughout the fish surface.

- Time histories of interesting time derivatives, like dp/dt, the pressure derivative with respect to time, in terms of local node data or instantaneous maxima throughout the surface of the fish.
- Time histories of the various forcing quantities that propagate the fish, viscous forces, buoyancy forces etc. This data may be used to understand why the fish is following a specific trajectory and to quantify the influence of the various driving terms of the motion equations.
- Time history of flow-induced loads on the fish, including forces and moments. These very important quantities are computed locally, on every node of the fish, as described in the previous sections, and can be interpreted either through local or integrated values for shear load, bending and torsion loads.

As revealing as time-histories can be, their usability is partially hindered by the fact that one usually has to deal with a huge number of individual graphs, since it is quite common to include dozens or even hundrends of fish in each VF run. A faster and more comprehensive way to short through all this data has been devised, a technique that is call "danger-zone maps". Such maps can be constructed, for a particular flow quantity, or flow-induced load acting on the fish, in the following manner: First we define a set of initial release locations, which are located on a plane upstream of the subsystem we wish to analyze. Fish are released from these initial locations and their trajectories through the subsystem are calculated using the VF model. Along the computed trajectory, we record certain quantities of interest, such as maximum pressure, or shear velocity, or strike velocity, etc. The so computed quantities are then assigned to the initial release location of the fish so that contour plots of each quantity can be constructed on the initial release plane. The resulting plots provide a wealth of information in a very compact manner and can be very helpful in assessing the fish friendliness of a given subsystem, as they can be used to identify the "danger zones" associated with this subsystem. Danger zones are defined as the areas upstream of the subsystem within which if fish find themselves are likely to experience, during their passage through the subsystem, a particular load that exceeds biological thresholds for survivability.

6. Representative results

In this section we present and briefly discuss a small representative set of computed results. We should emphasize that this section is intended only to demonstrate the ability of the VF model to calculate fish trajectories through every component of a typical hydraulic powerplant. Extensive applications of the model over a range of operating conditions and comprehensive analysis of the computed results will be the primary focus of the second phase of this work. The flowfields used to obtain the subsequently presented results were obtained by Voith's CFD group using AEA's TASCFLOW commercial software (TASCFLOW Users Manual, 1998). Flow simulations were carried out for most subsystems of the Wanapum dam at a single operating condition.

Since it has been shown in the literature that shear can be a critical, injury-inducing factor at the head-gills-eyes region, a focal point of the results presented is the determination of shear on or near one of the fish gills. Shear forces are quantified in terms of the magnitude of the average, over the fish body, relative velocity between the fish and the ambient flow, or similarly, by the relative velocity between the fish and the ambient flow at one particular point on the fish body. We also monitor, along computed trajectories, two additional quantities: i) pressure variation; and ii) acceleration magnitude, which serves as an estimate of the total force acting on the fish body. Finally, for those trajectories that involve impact, strike or other wall contact events, the impact velocity of the fish is recorded. All results are dimensional (S.I. unit system) and correspond to the full-scale hydro turbine. A virtual fish of 0.2m x 0.05m x 0.03m principal dimensions, with a weight of approximately 0.2 kg, discretized using 121 surface elements, has been used for all the simulations of this section.

6.1 Description of the Wanapum CFD Model

The trajectories of virtual fish were calculated in a region spanning the intake bays of the semispiral case to the draft tube cone. This section of the machine was divided into two subsystems: one containing the intake, scroll, stay vanes and guide vanes, and the other including the runner and the draft tube cone. The first subsystem was modeled with approximately 300,000 nodes for a flow rate of 17,000 cfs. The second subsystem is that of the five-blade runner at a tilt of 31.35 degrees and includes the hub ramp structure as well as gaps at the hub and periphery. In the CFD analysis, a single blade was modeled with periodic boundaries, thus utilizing the assumption that the velocity field in the vicinity of each blade is the same. The assumption of periodic flow allows us to carry out our computations for a single, blade-to-blade passage using a very fine grid resolution. A total of approximately 402,000 nodes are used to discretize the blade passage, which corresponds to a grid for the complete runner consisting of more than two million nodes. The flow rate for this component is 11,000 cfs, with the inflow velocity angles determined from a separate analysis of the stay vanes and wicket gates at this operating condition.

6.2 Passage through the Intake/Scroll/Stay-Vanes/ Guide-Vanes Subsystem

In order to carry out a comprehensive investigation of the fish passage characteristics within the first subsystem of the powerplant, a total of 300 fish were evenly distributed among the three intake bays and tracked through the scroll and vanes. A sample of 18 fish trajectories is presented in figure 5. Even for the six trajectories per intake bay depicted in this figure, several qualitative trends can be identified. For instance, fish entering from the left intake bay tend to span most of the scroll with their trajectories and enter the stay vanes from the back part of the scroll. Also, several fish entering from the middle and right bays (for the distribution presented in figure 5) appear to come in contact with either the stay vanes or the guide vanes at the front part of the scroll section.

A number of additional features of the fish passage can be elucidated by considering a different set of release locations. Figure 6 shows a horizontal, constant-depth, distribution of initial positions. It is seen that fish entering from the left-most part of the left bay appear to scrape the rear wall of the scroll, whereas fish entering from the right-most part of the right bay, consistent with the behavior discussed in the previous paragraph, seem to come in contact with the vanes arrays. We have found, however, that these fish strike the rear part of the scroll ate relatively small angles and with rather small velocities (typical impact velocities in this part of the system were found to be less than 1m/s). We do not anticipate, therefore, that the scrape events indicated by the model in this region would result in significant injuries. Instead, fish would probably bounce back in the flow, with minor, if any, injuries and continue their trajectories.

As we have already discussed above, the VF model can record specific quantitative information regarding the local flow quantities encountered by fish along their trajectories. Figure 7 shows the pressure and shear velocity at the left gill and also the average acceleration of fish No 4 of figure 6. A mild increase of the pressure (which is obviously hydrostatic) is followed by a sharp drop, which corresponds to the passage through the vanes arrays. The sudden change of direction of the fluid in that region is reflected in the rapid increase of the shear velocity during the last few seconds of this trajectory and also in the time-lagged acceleration that the fish undergoes.

Another interesting behavior is recorded in figure 8, which depicts the time histories of fish No 5 of figure 6. This particular fish clears the first array of vanes but comes in contact with the second set. Its fate is reflected in the time-records presented in figure 8. The mild hydrostatic pressure variation that occurs as the fish is carried from the inlet to the vanes is followed by a short but sudden drop that corresponds to the acceleration within the first cascade passage. This acceleration is evident both in the shear velocity and in the total fish acceleration signal of figure 8.

Presenting information in the manner shown in figs. 7 and 8, although helpful in analyzing the behavior of individual fish, does not enhance our understanding of the global fish passage characteristics. For that reason, and to facilitate an in depth analysis of fish passage, the "danger-zone" maps technique, as described in the previous chapter, is used.

To demonstrate the concept of the danger-zone map, we employ it to evaluate the global fish passage characteristics for the subsystem consisting of the stay and guide vanes. We chose not to apply this technique to the intake/scroll subsystem as this is the region within which we anticipate the accuracy of the VF model to deteriorate considerably due to the overall slow flow velocities and the potential for significant fish free-will effects (which have not been incorporated in the model yet). The initial release plane and the fish release locations on this plane are shown in figure 9. Approximately 220 fish are introduced along a segment outside of the vanes periphery, that spans 1/5 of the total scroll circumference. This was adopted in order to have an initial fish release distribution compatible with that used in the subsequent section for the analysis of the runner subsystem. Few typical subsets of fish trajectories are shown in figures 10 (for a horizontally oriented subset) and 11, (for a vertically oriented one)--the jagged shape of the trajectories in these figures is due to the fact that, to reduce storage requirements, fish locations are plotted only every two hundred time steps. Figure 12 shows the "danger-zone" map for the maximum shear velocity corresponding to the initial locations shown in figure 9. The vertical strips of high shear velocity at the top part of this figure correlate with pairs of stay vanes-wicket gates. The diagonal stripes at the lower part, on the other hand, should be attributed to the local flow conditions within the scroll, where the flow encounters the vane arrays at an angle and not horizontally as the flow at the top part does. Similar patterns are observed in figures 13 and 14, which depict the maximum acceleration and impact velocity. For this latter figure, a zero impact velocity implies that no impact took place. As seen in figure 14, the majority of fish clears this subsystem without striking walls. Even those that do strike, do so at relatively low velocities and only a small percentage of fish could possibly experience mild injury-inducing strike events (for velocities greater than 3m/s). Similar quantitative results can also be deduced from the shear-velocity map--maximum shear velocities nowhere exceed 3.2 m/s.

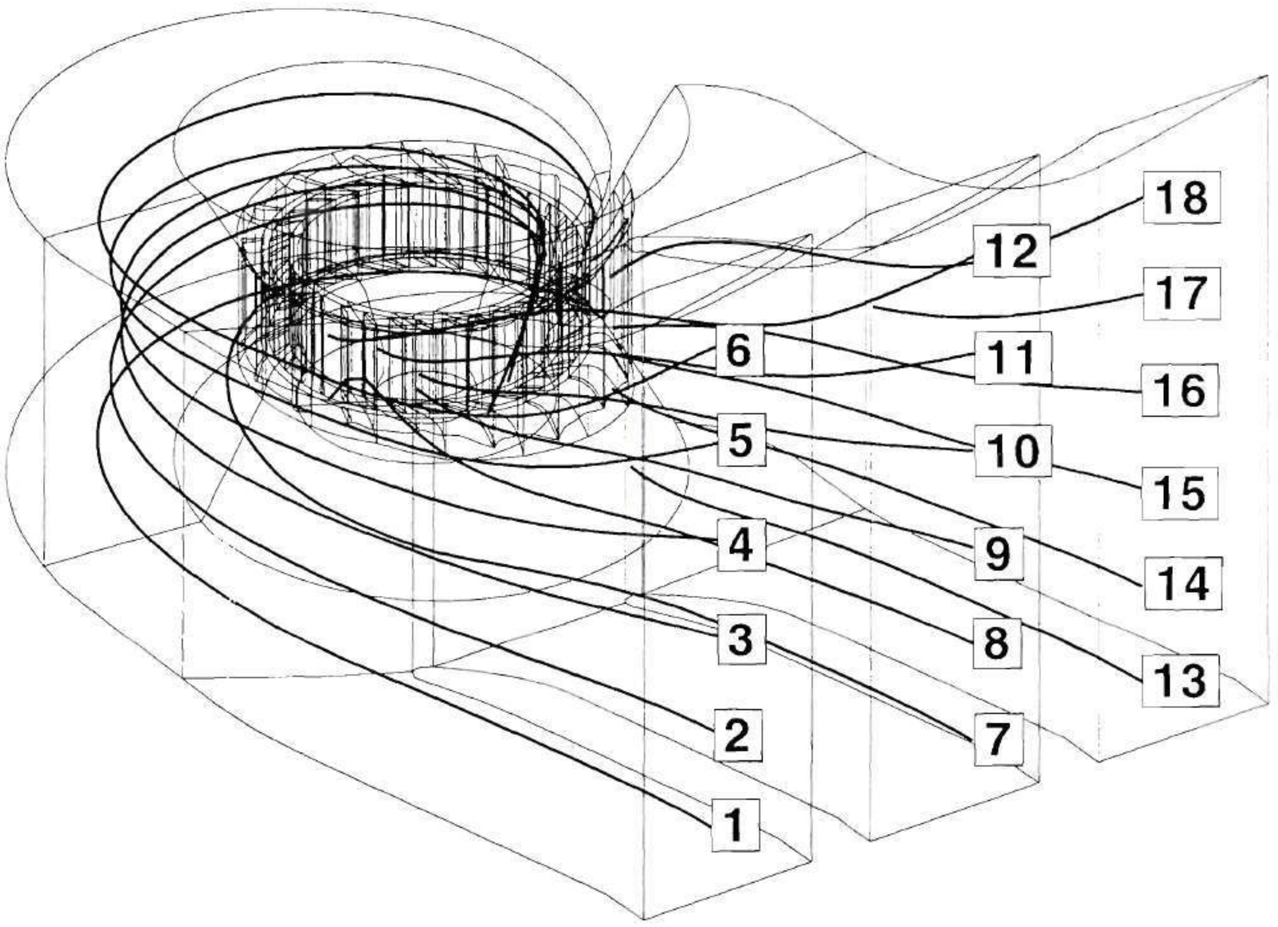


Figure 5. Typical fish trajectories for vertical arrays of release locations for the inlet-scroll-vanes subsystem

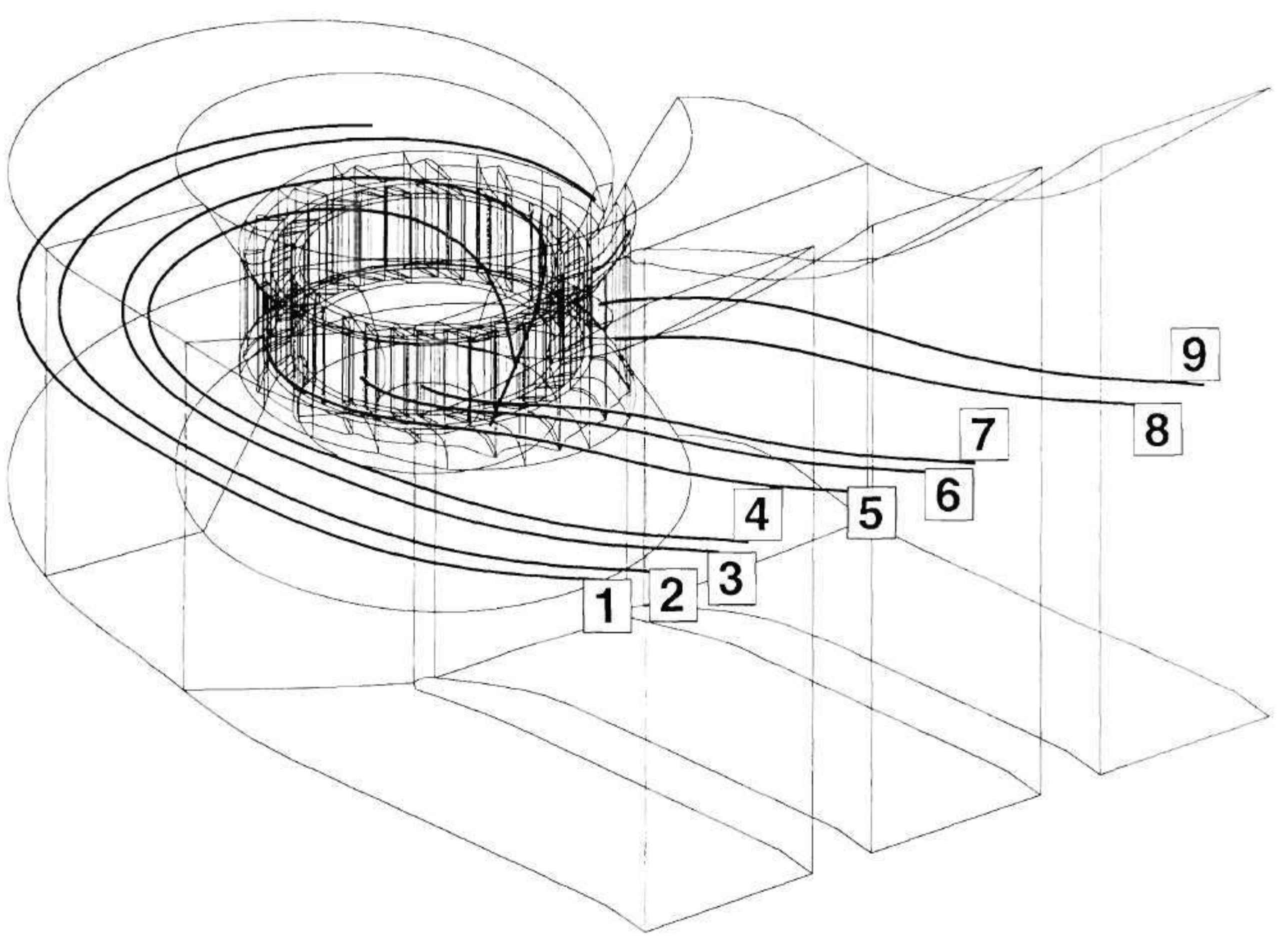


Figure 6. Typical fish trajectories for a horizontal array of fish release locations for the inlet-scroll-vanes subsystem

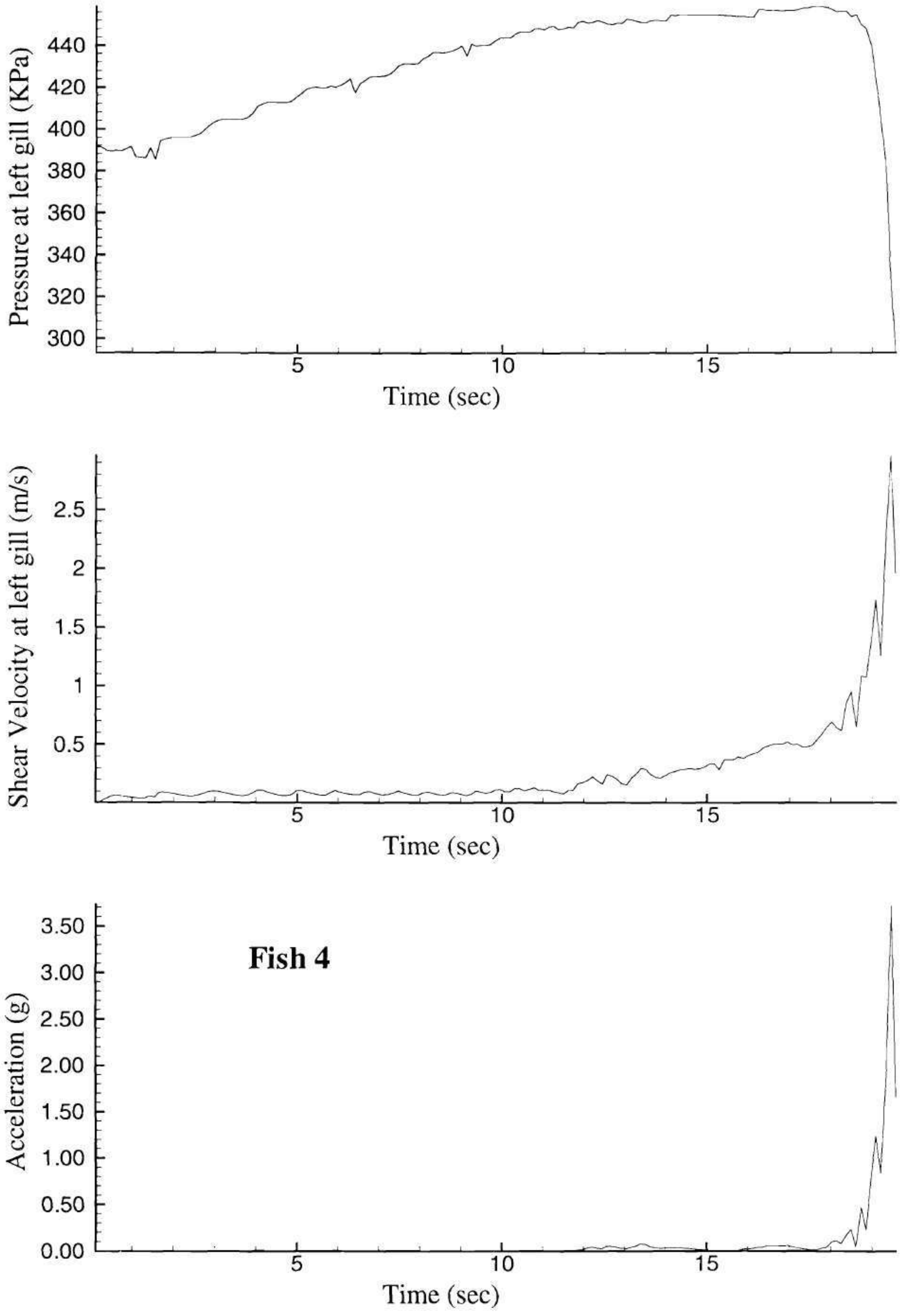


Figure 7. Pressure, shear velocity and acceleration time histories for fish No 4 of figure 4

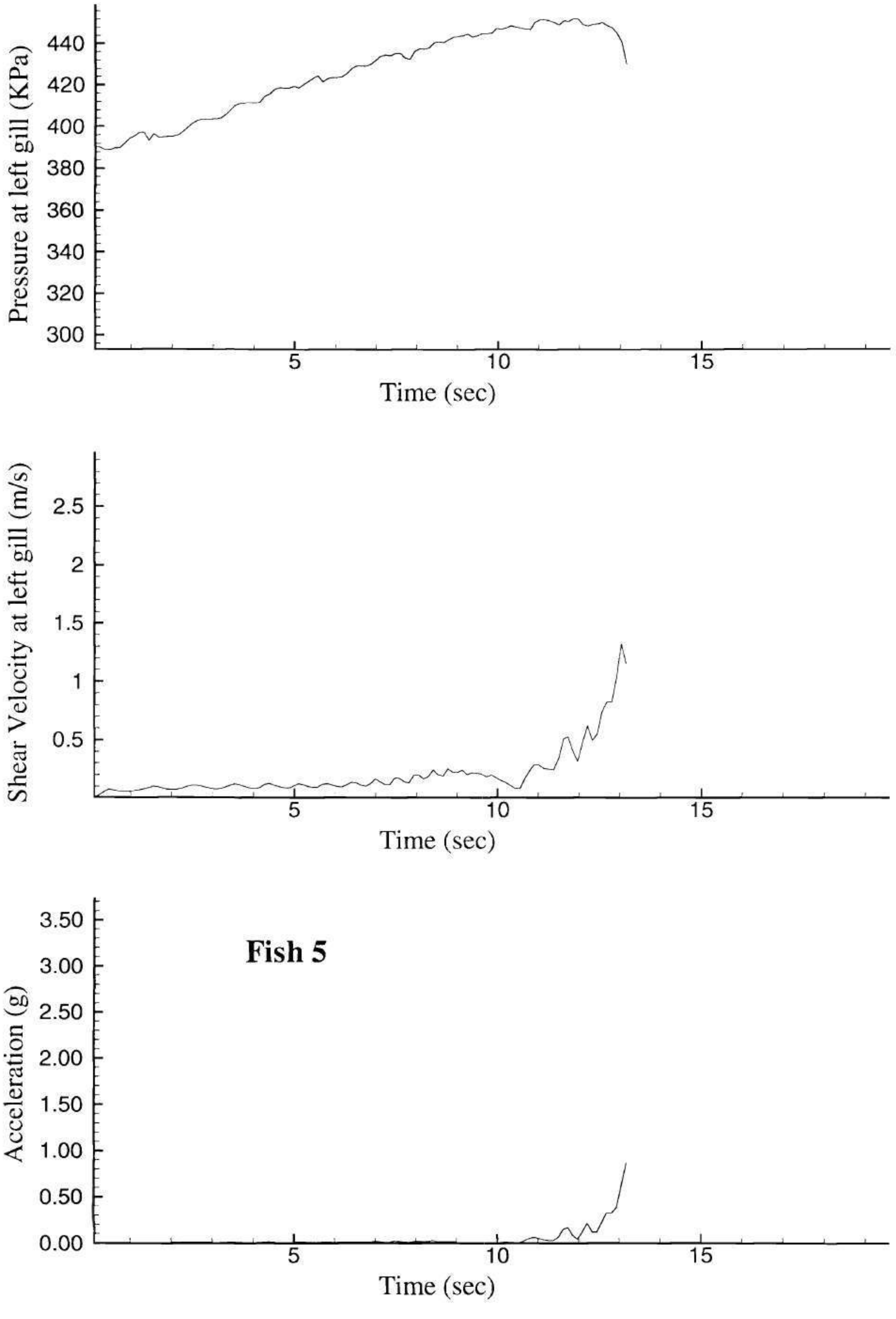


Figure 8. Pressure, shear velocity and acceleration time histories for fish No 5 of figure 4

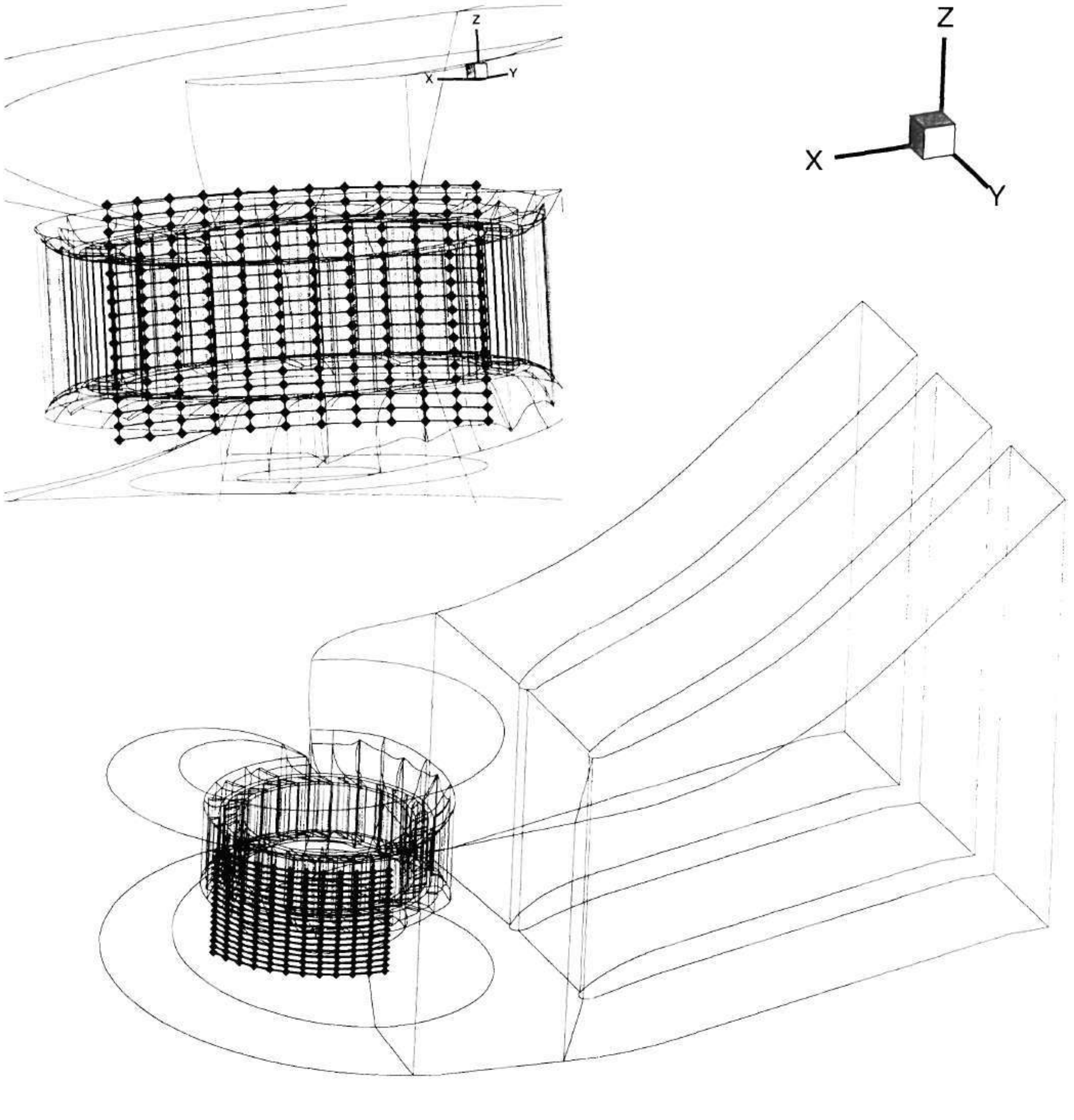


Figure 9. Initial fish locations for the vanes region

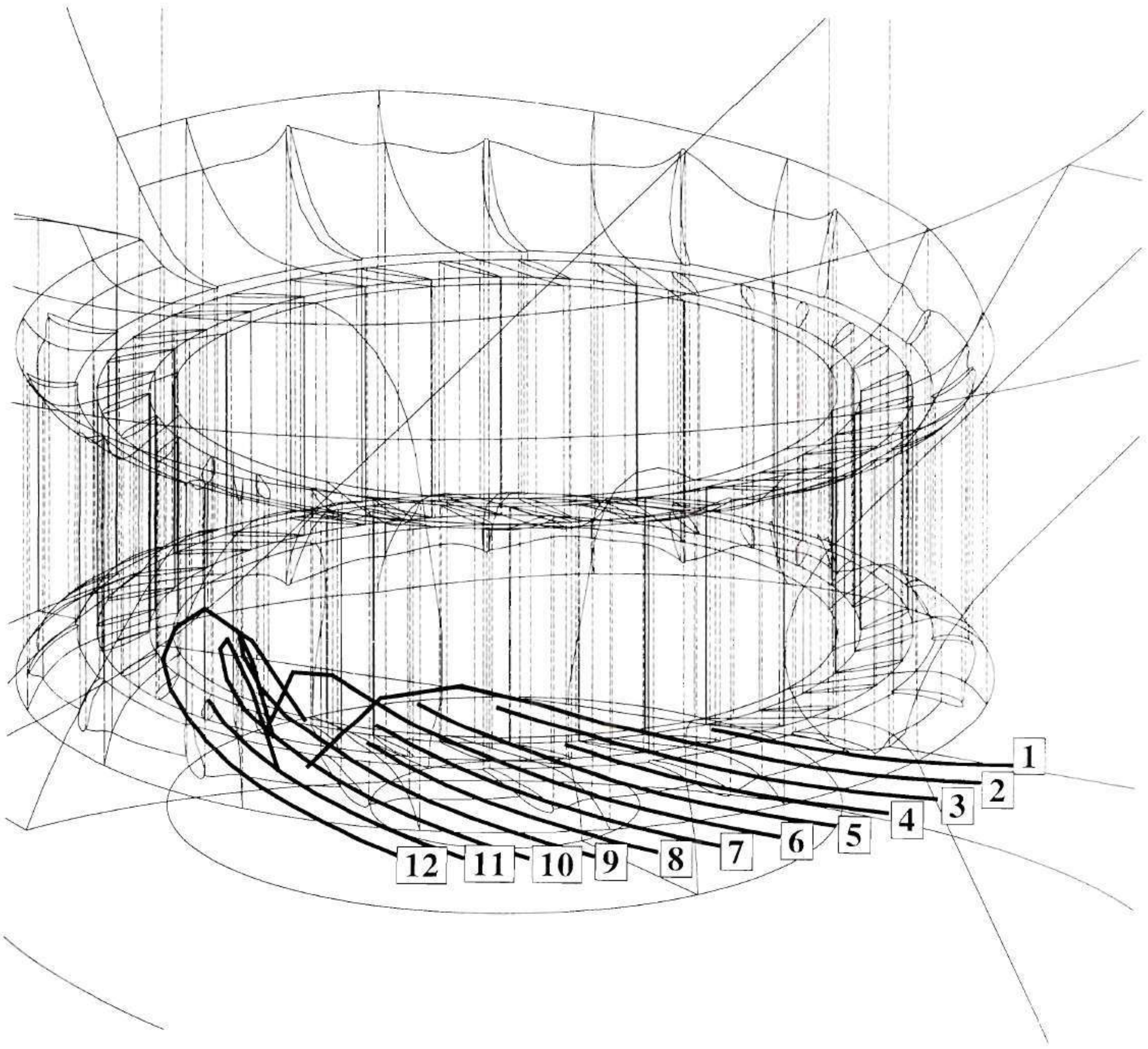


Figure 10. Typical fish trajectories for a horizontal array of release locations for the vanes region

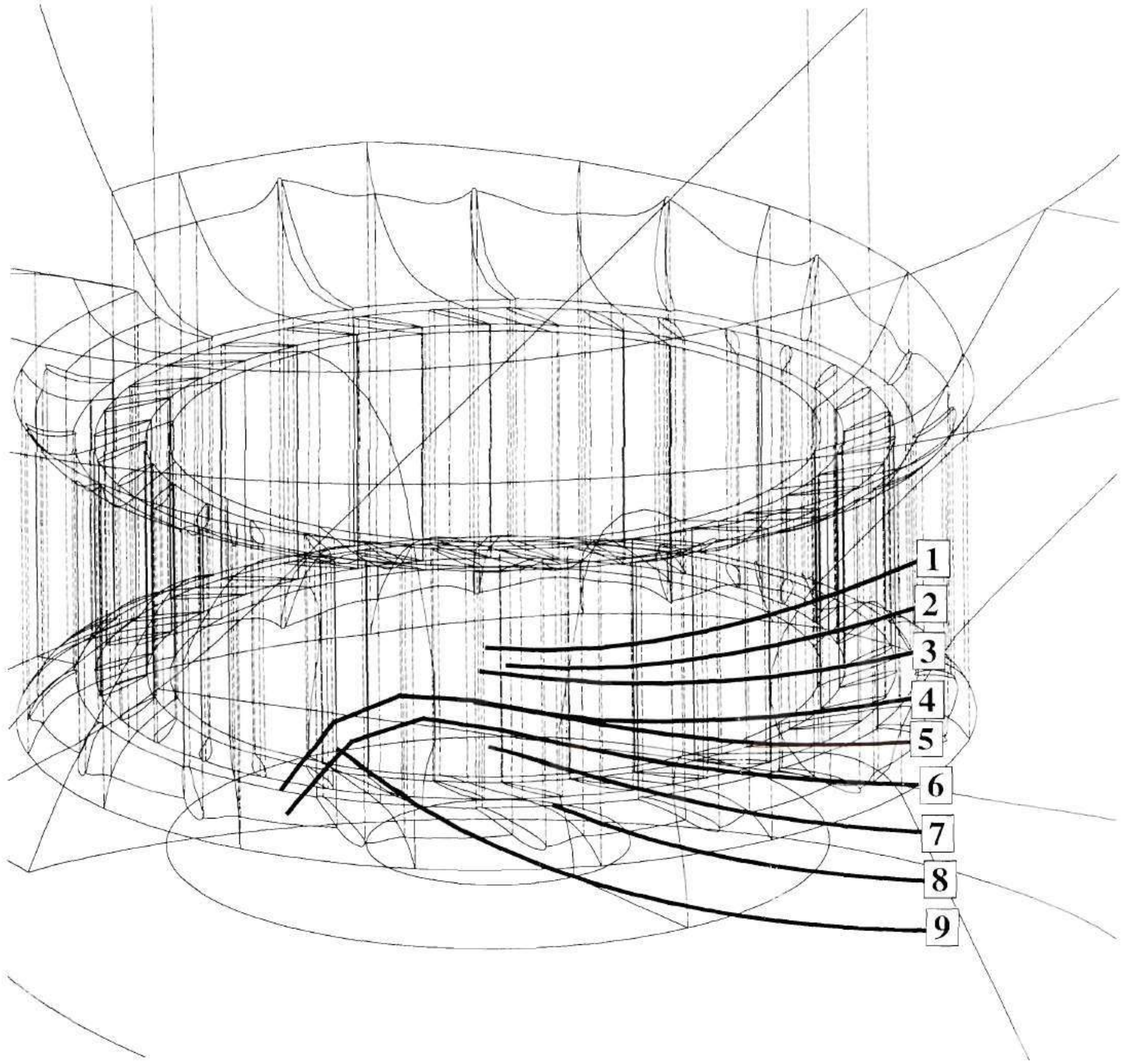


Figure 11. Typical fish trajectories for a vertical array of release locations for the vanes region

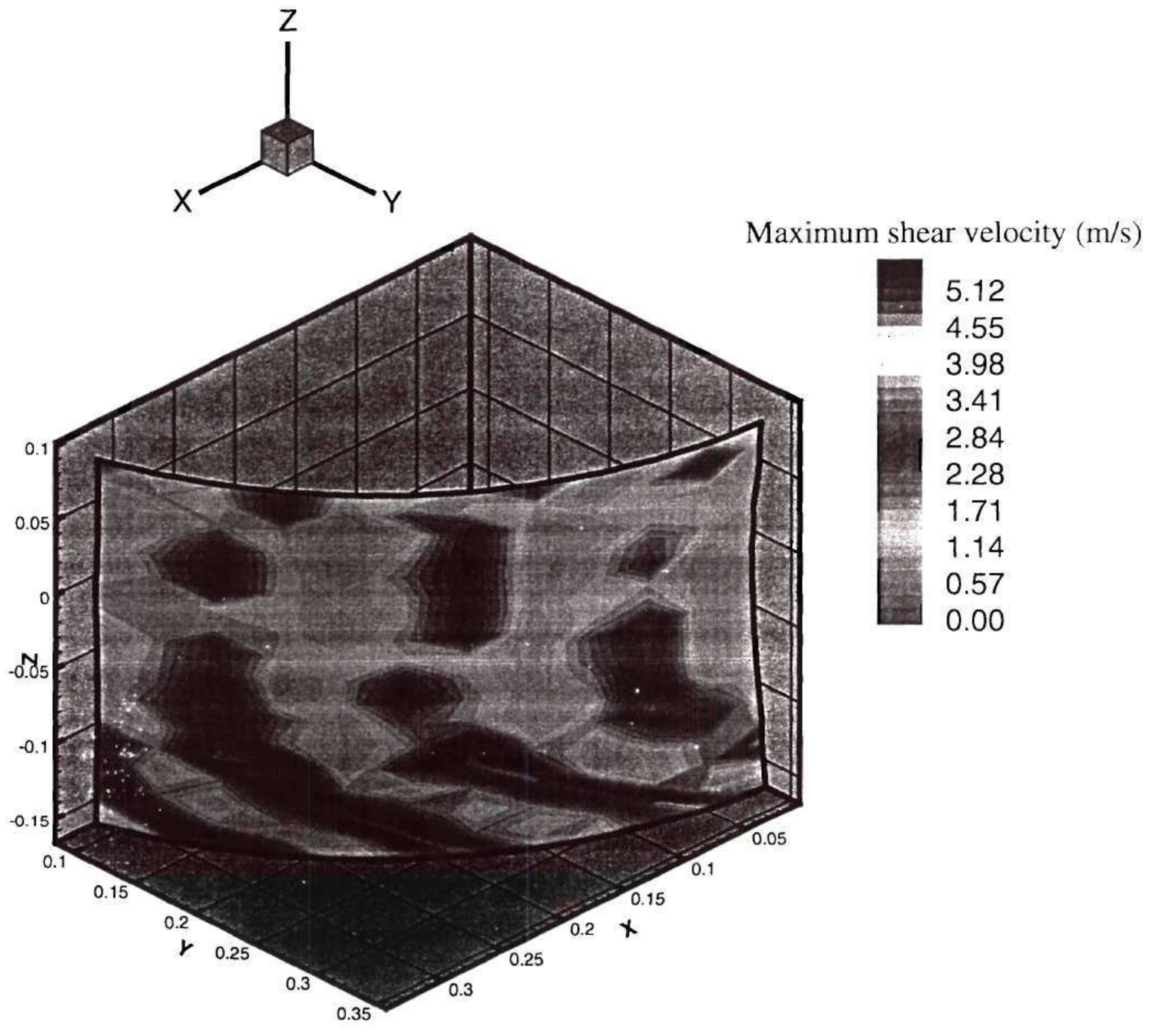


Figure 12. Maximum shear velocity iso-contours for the initial fish release locations of figure 9

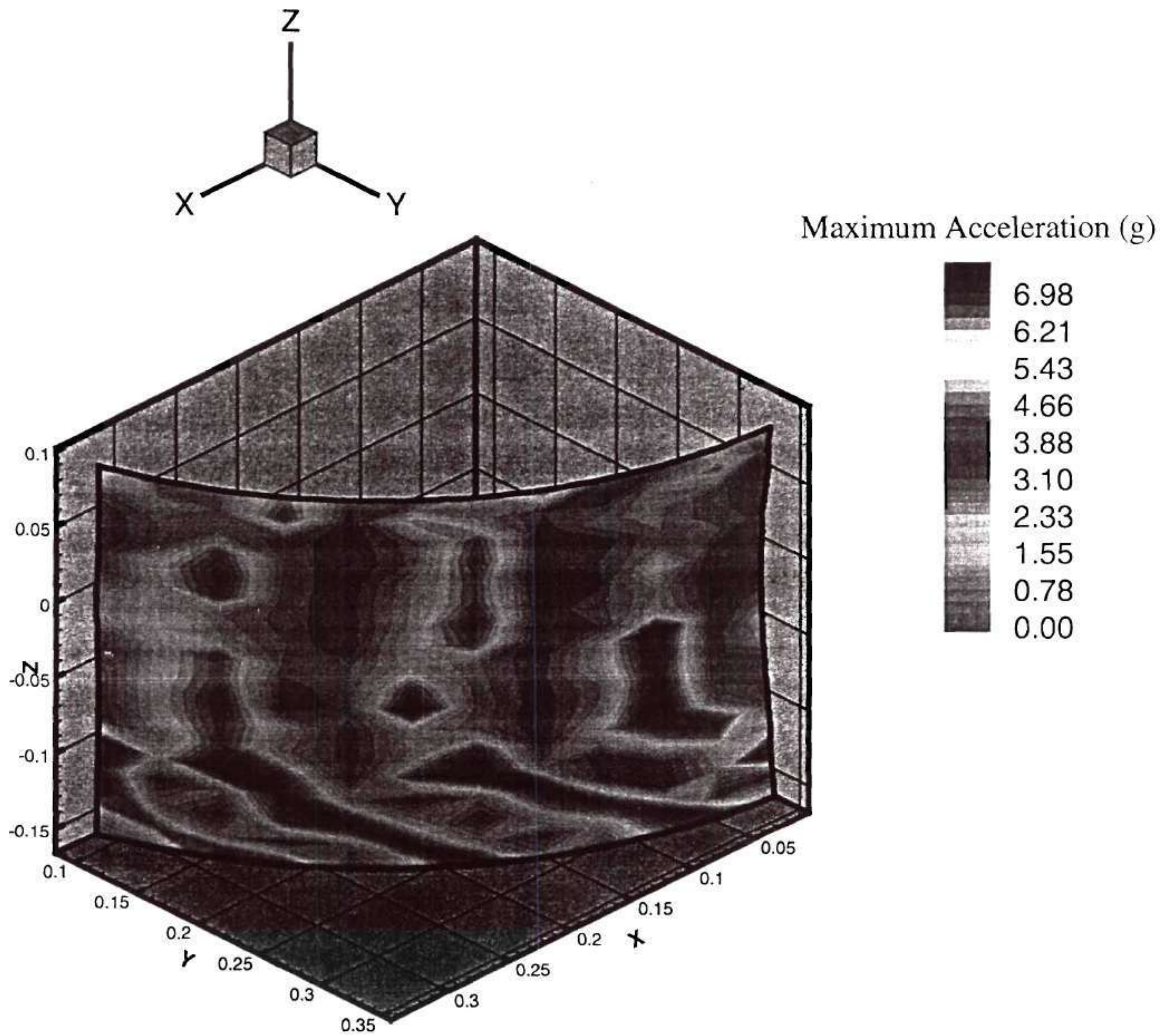


Figure 13. Maximum acceleration iso-contours for the initial fish release locations of figure 9

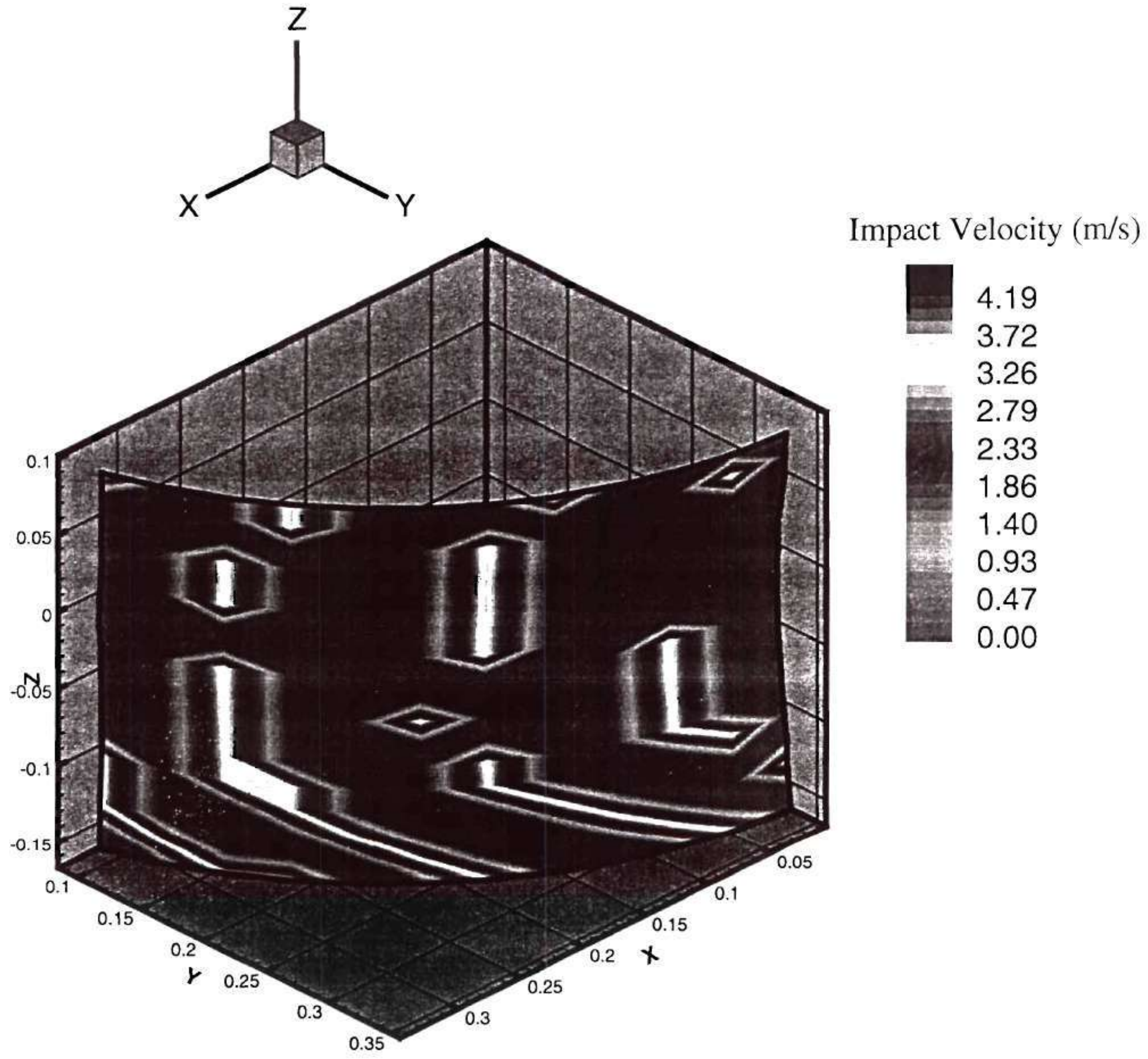


Figure 14. Impact velocity iso-contours for the initial fish release locations of figure 9

6.3 Runner Fish Passage Characteristics

A similar analysis as the one presented above is also performed for the runner. Since the runner CFD solution is periodic for the five blades, we can adequately analyze the fish behavior in this component by introducing fish at a section spanning 1/5 of the periphery of the runner. Again, a systematic distribution of approximately 100 fish release locations is used to fully cover this part of the periphery, figure 15.

Figure 16 depicts trajectories originating from a subset of the aforementioned release locations, oriented along the runner axis. Out of the seven trajectories presented in this figure, the three near the top seem to come in contact with the runner blades, whereas the four near the bottom clear the runner and exit through the outflow plane to the draft tube ring. Figure 17, depicting a horizontal initial location distribution, shows four trajectories clearing the blade, along with one that scrapes tangentially the blade trailing edge.

Figures 18 and 19 show the time records for two fish of figure 17, namely fish 1 and 3. Although the trajectory for fish 1 strikes the blade trailing edge, it does reveal some typical features of runner fish passage, such as the rapid pressure drop along the blade and the intense acceleration the experienced by the fish. The same overall features can also be identified in figure 19. This figure also reveals that the fish passes through a relatively short zone of pressure recovery and near-zero acceleration after it clears the blades and before reaching the draft tube ring.

The danger-zone maps for the maximum acceleration, shear-velocity, and strike velocity, for the set of initial conditions shown in figure 15, are shown in figures 20, 21, and 22. All three plots exhibit a diagonally oriented high-intensity zone, which should be linked to the orientation of the runner blade. Comparisons with the danger-zone maps for the vanes subsystem (see figures 12, 13, 14) indicate that fish passing through the runner are likely to be exposed to considerably higher levels of shear and strike velocities, which could induce injuries and possibly mortality. That is, the relatively uneventful passage of fish through the upstream components is accompanied by a potentially dangerous passage through the runner blades. We should emphasize, however, that insofar as both shear and strike are concerned only a small percentage of release locations will expose fish to potentially lethal levels.

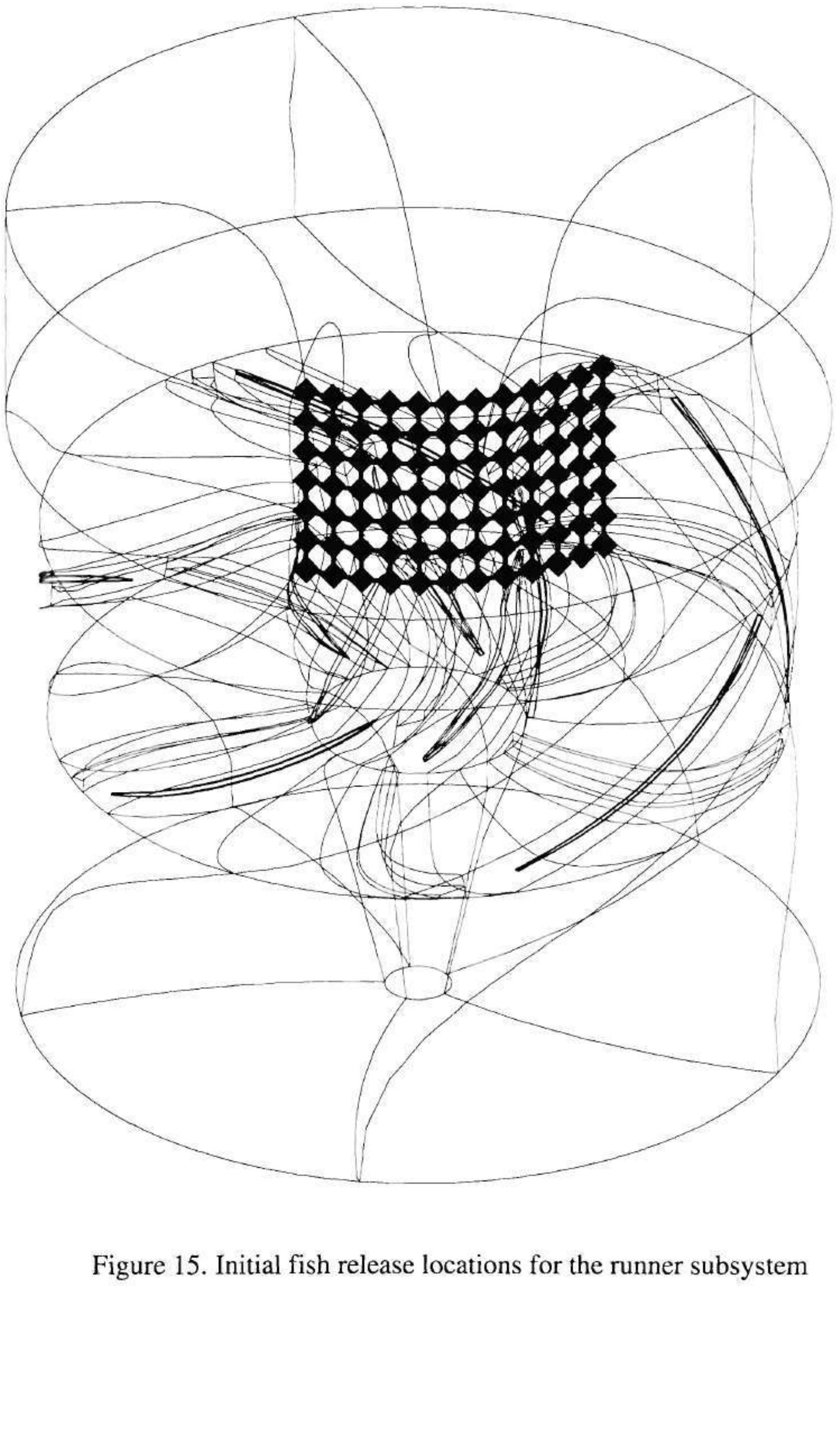


Figure 15. Initial fish release locations for the runner subsystem

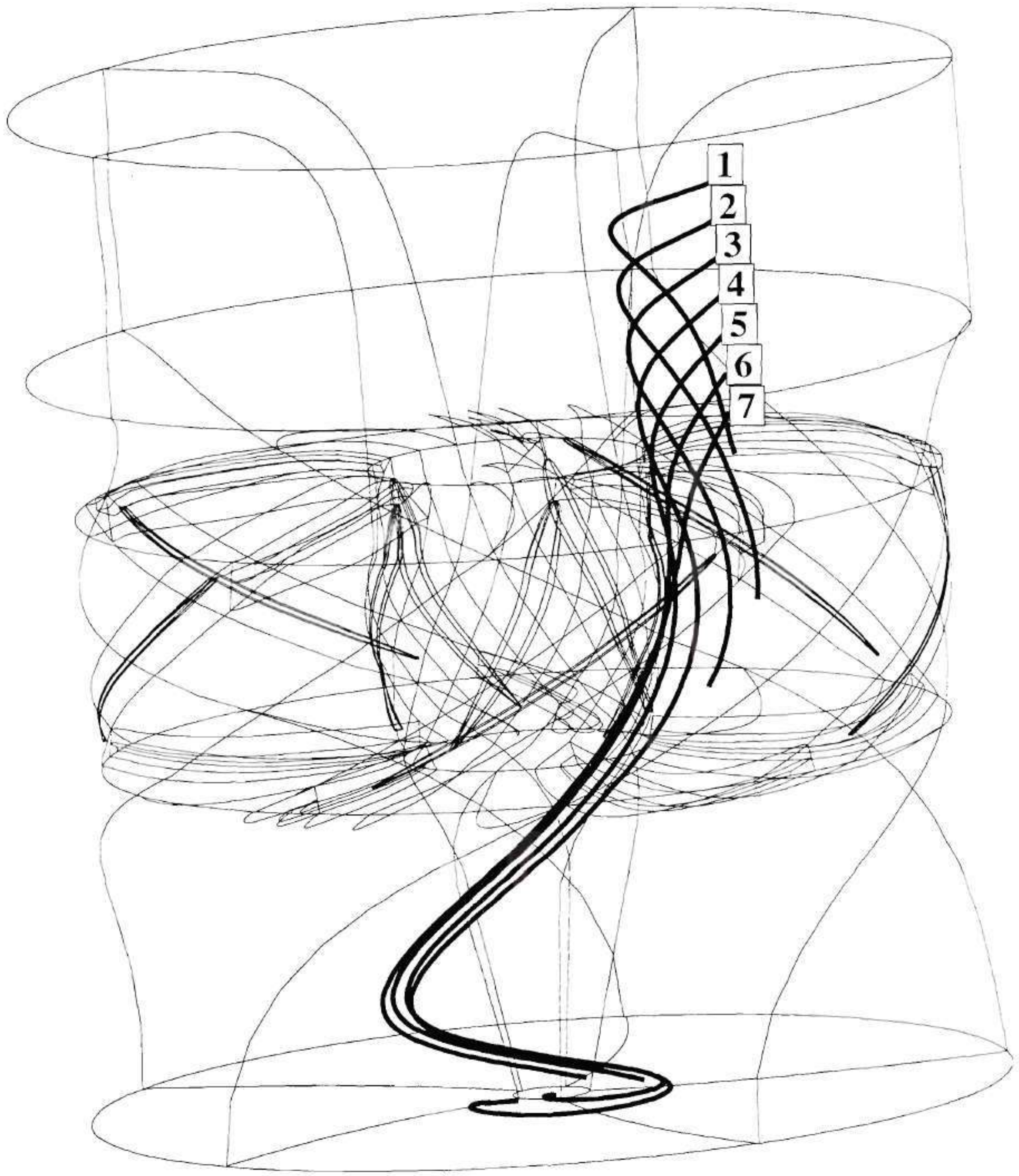


Figure 16. Typical fish trajectories for a vertical array of release locations for the runner subsystem

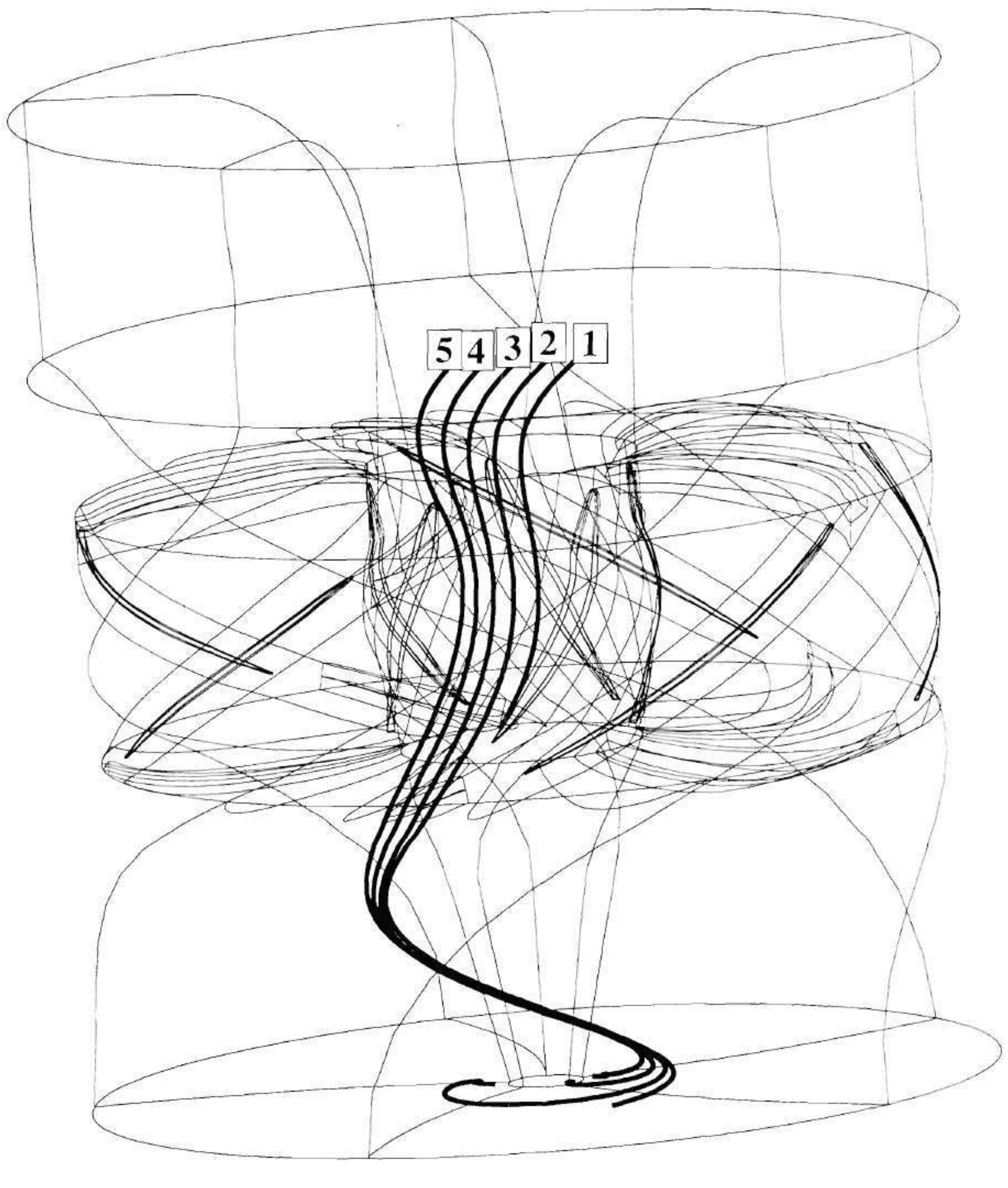


Figure 17. Typical fish trajectories for a horizontal array of release locations for the runner subsystem

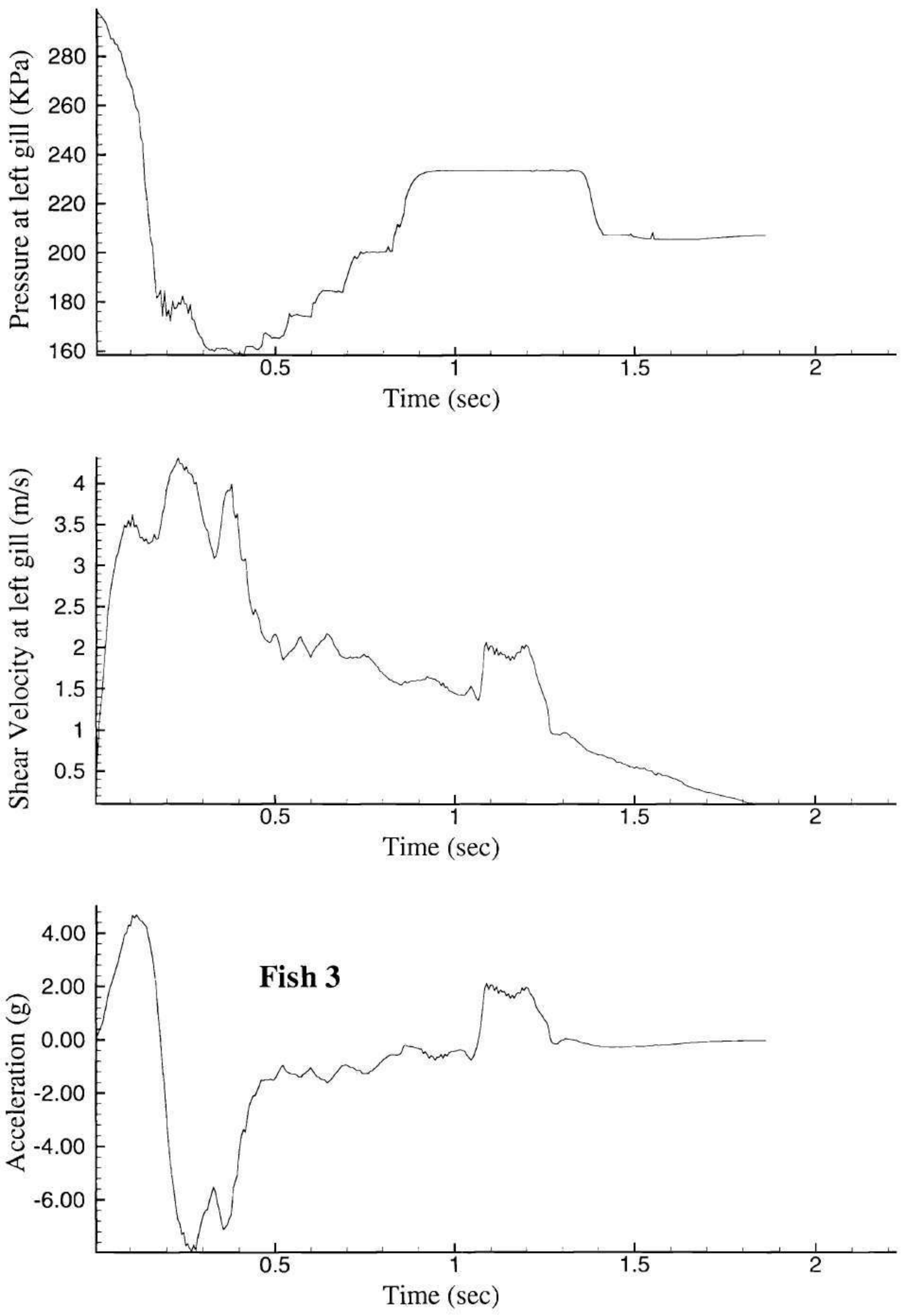


Figure 19. Pressure, shear velocity and acceleration time histories for fish No 3 of figure 17

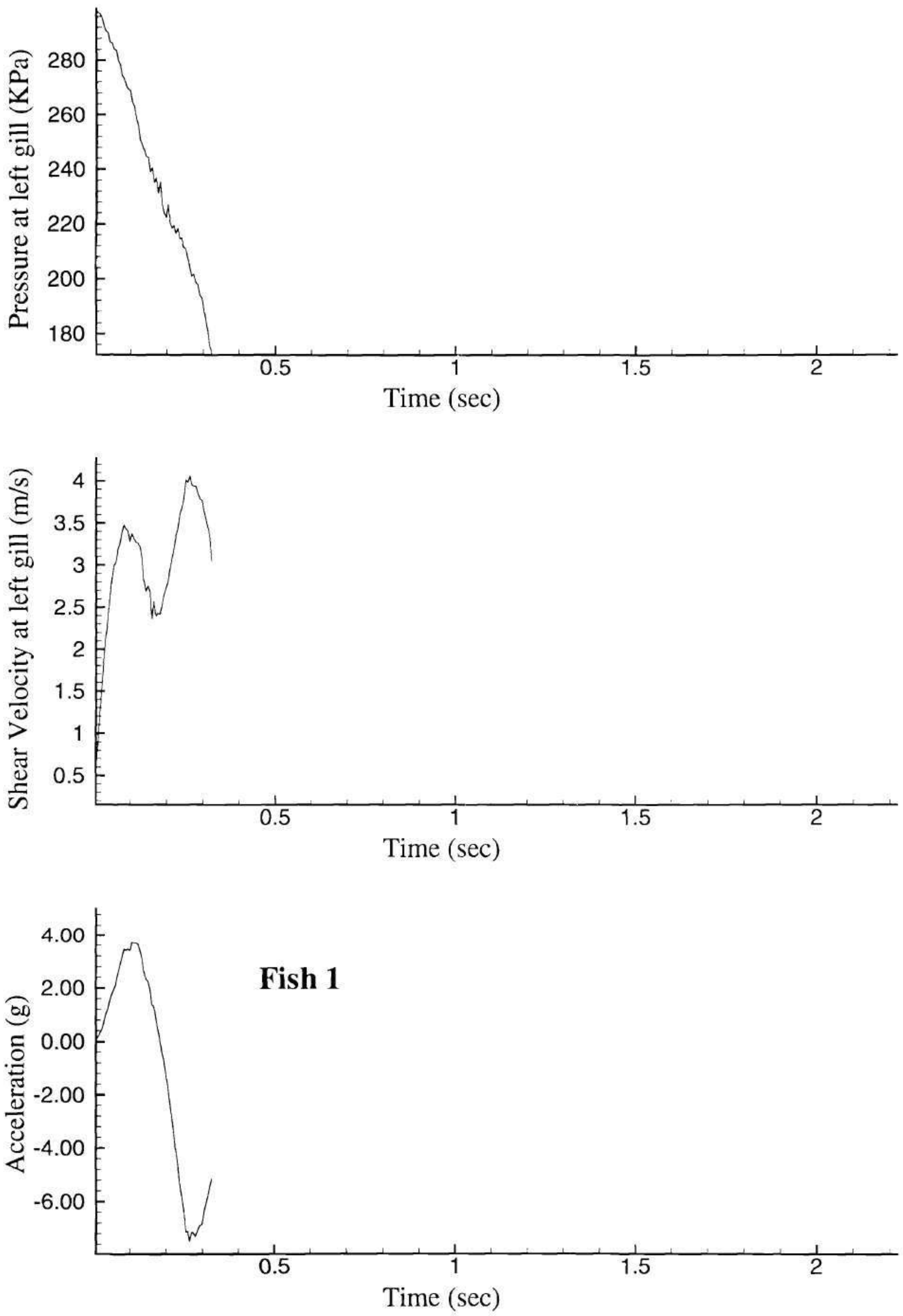


Figure 18. Pressure, shear velocity and acceleration time histories for fish No 1of figure 17

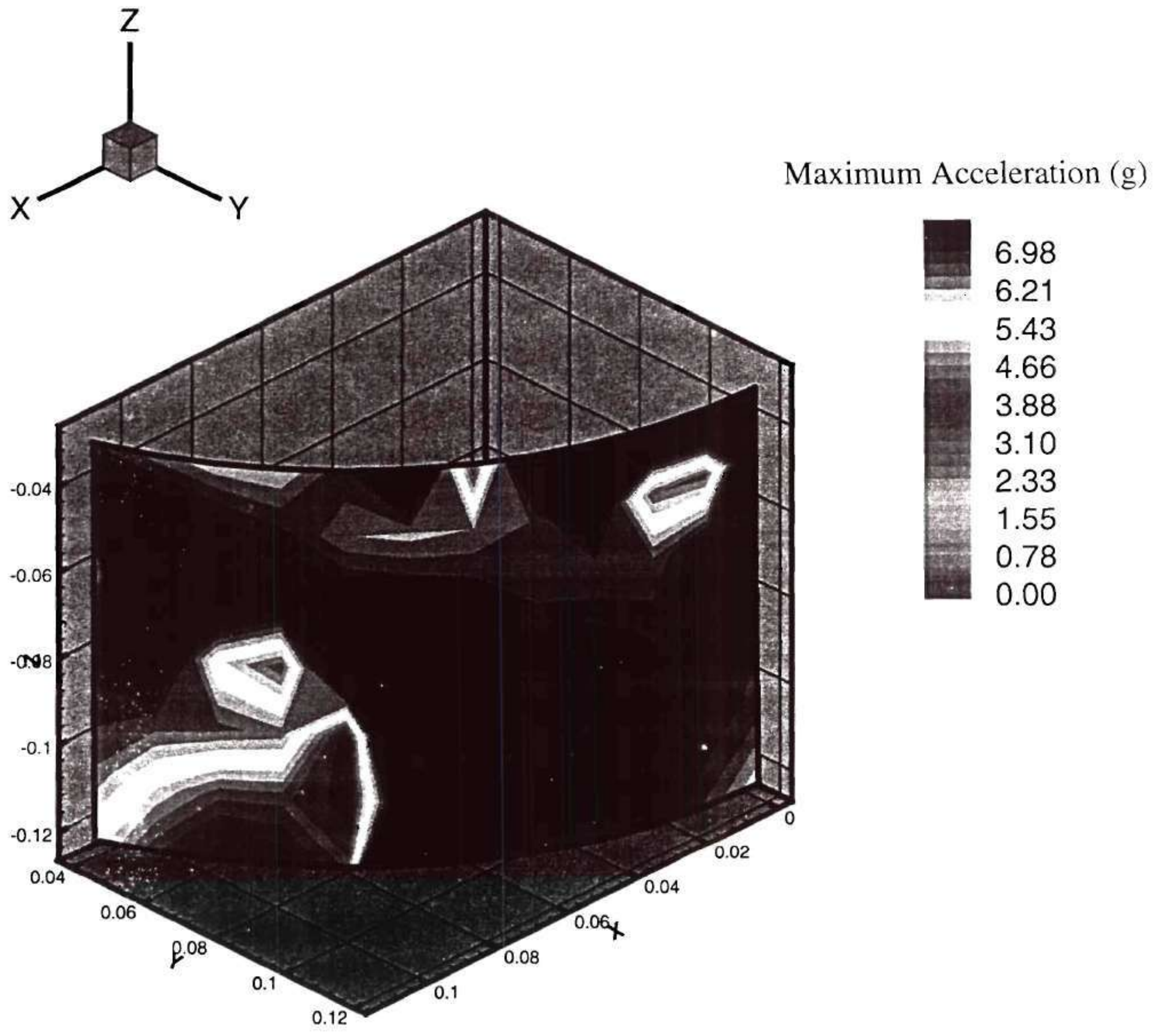


Figure 20. Maximum acceleration iso-contours for the initial fish release locations of figure 15

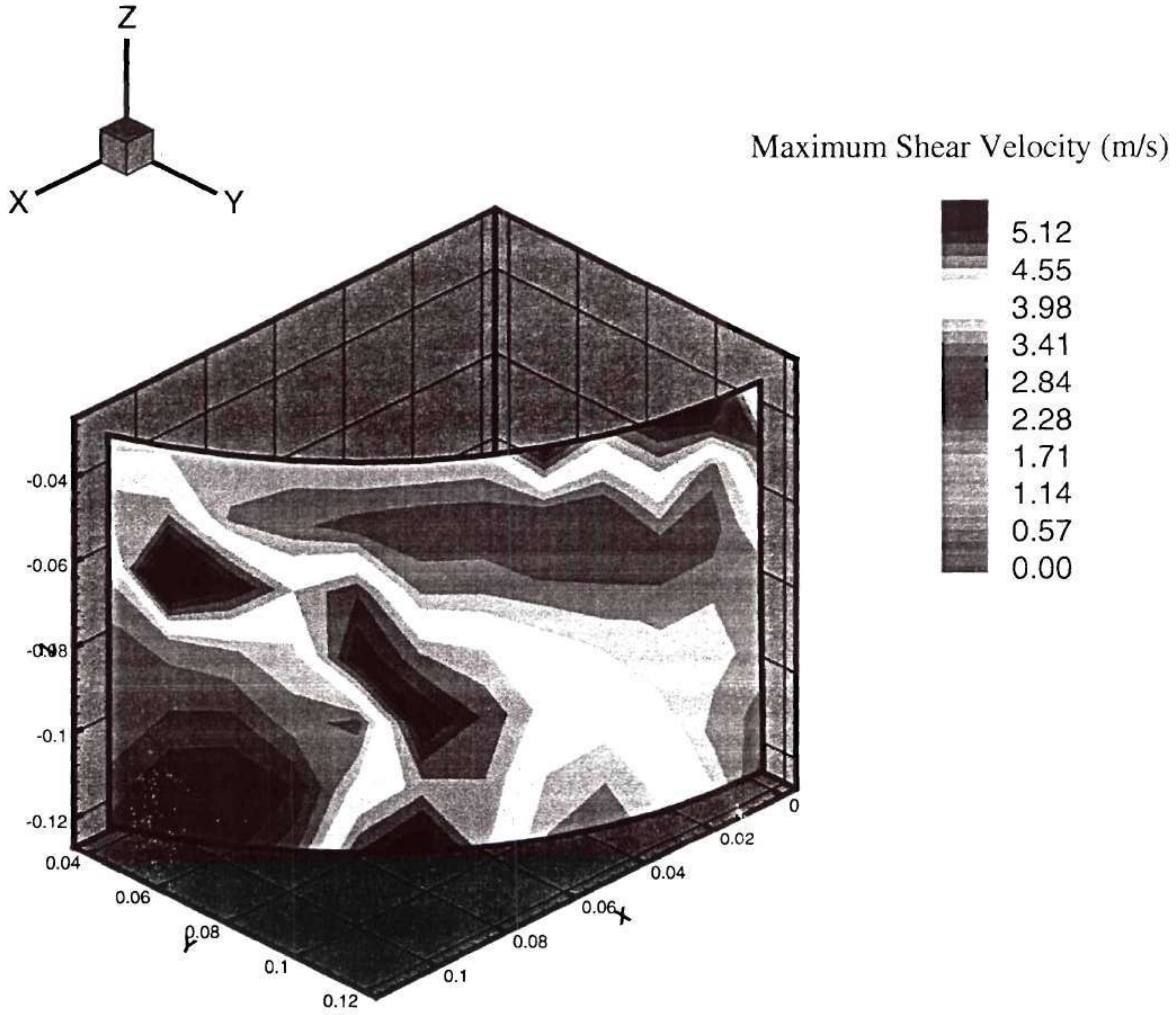


Figure 21. Maximum shear velocity iso-contours for the initial fish release locations of figure 15

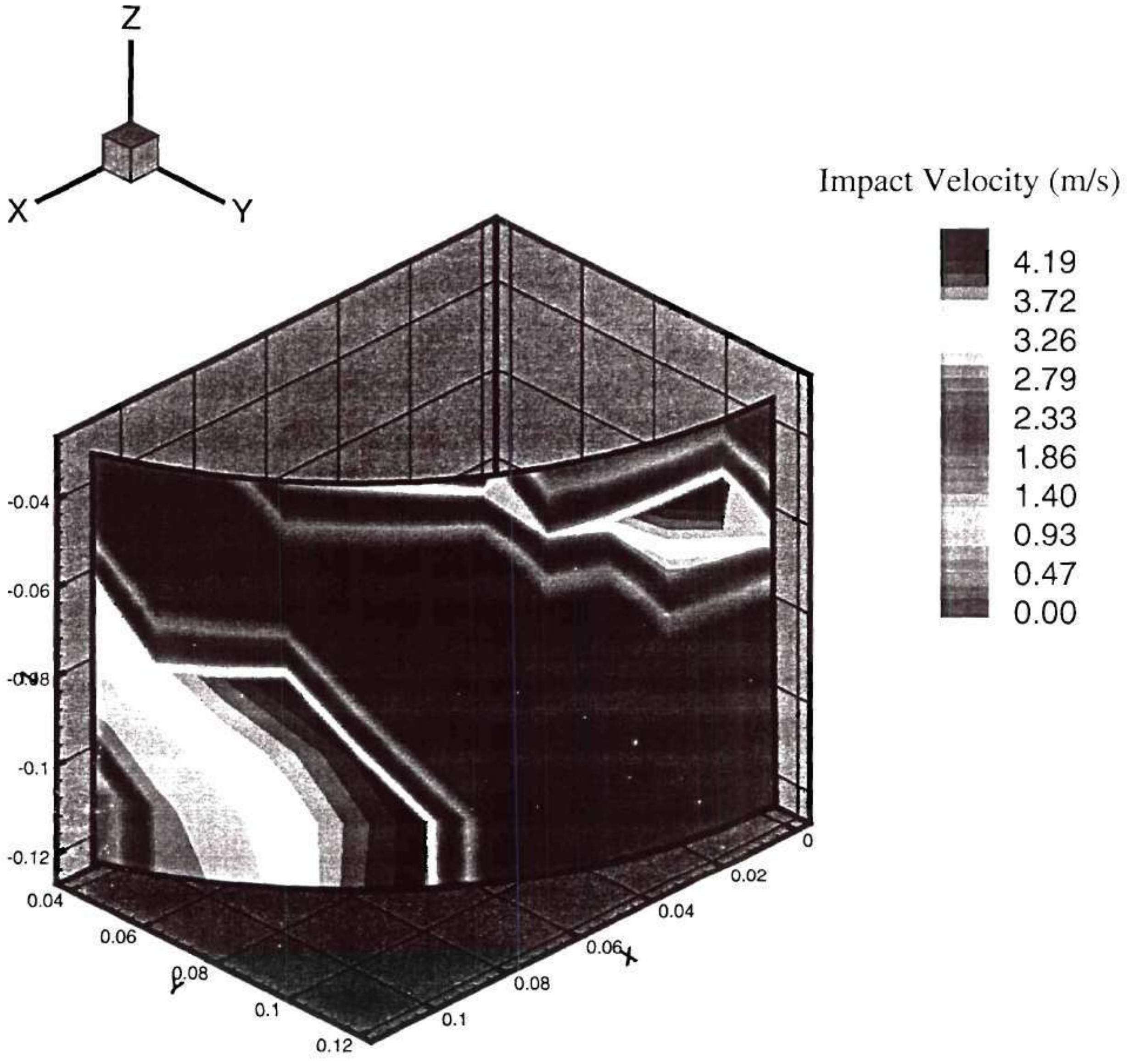


Figure 22. Impact velocity iso-contours for the initial fish release locations of figure 15

7. Enhanced visualization capabilities of the VF model

As we have already discussed, the VF method simulates the three-dimensional trajectories of fish-like bodies through realistic turbine geometries. Here we demonstrate how the output of the model can be utilized to visualize fish passage in a manner that allows for a better understanding of the interaction between the fish-like body and the machine components.

In the complex, multi-element environment of the Wanapum plant, figure 23, we visualize several events of interest. Figure 24, for instance, depicts a fish just before its impact with a vane. Figure 25 shows a rake of streamlines, along with two fish trajectories that clear the vane arrays and enter the runner. Figure 26 shows the same situation, viewed from the turbine shaft. Figure 27 demonstrates an additional capability of *enhanced visualization* options embodied in the VF model, i.e. that of color-coding the surface of the fish-like body according to the values of a quantity of interest, pressure in this case. Figures 28 and 29 present similar visualizations for the runner subsystem.

We should emphasize that the above results are only indicative of the capabilities of the model. The user can take advantage of these capabilities in order to better correlate fish trajectory specifics with features of the flow and elaborate on the understanding of the effect certain flow mechanisms have on fish passage. Moreover, even more advanced visualization techniques, like immersive 3D stereoscopic animation, are well within the capabilities of the VF model since all the necessary information is computed and stored.

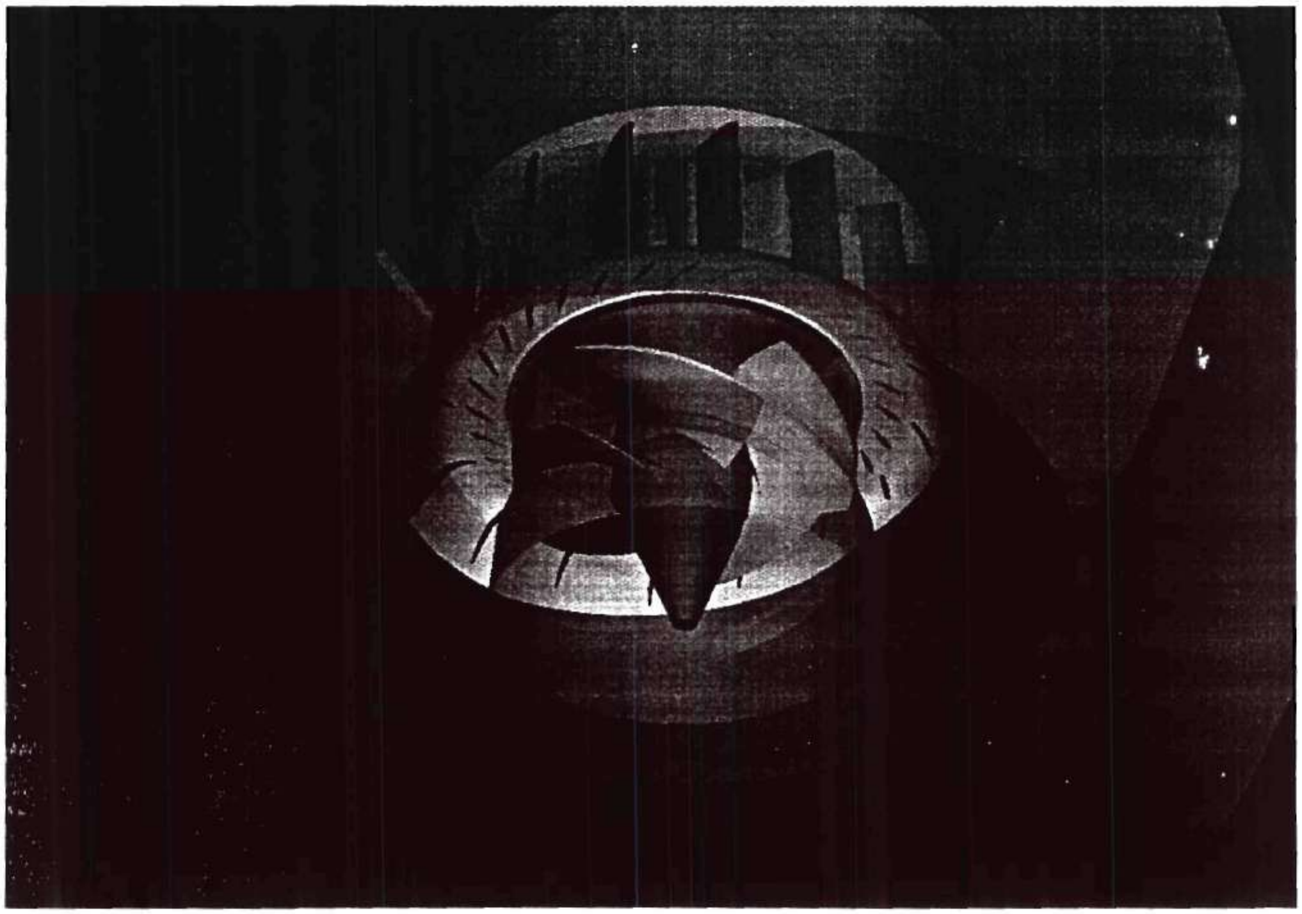


Figure 23. Global view of the simulated environment for the Wanapum project

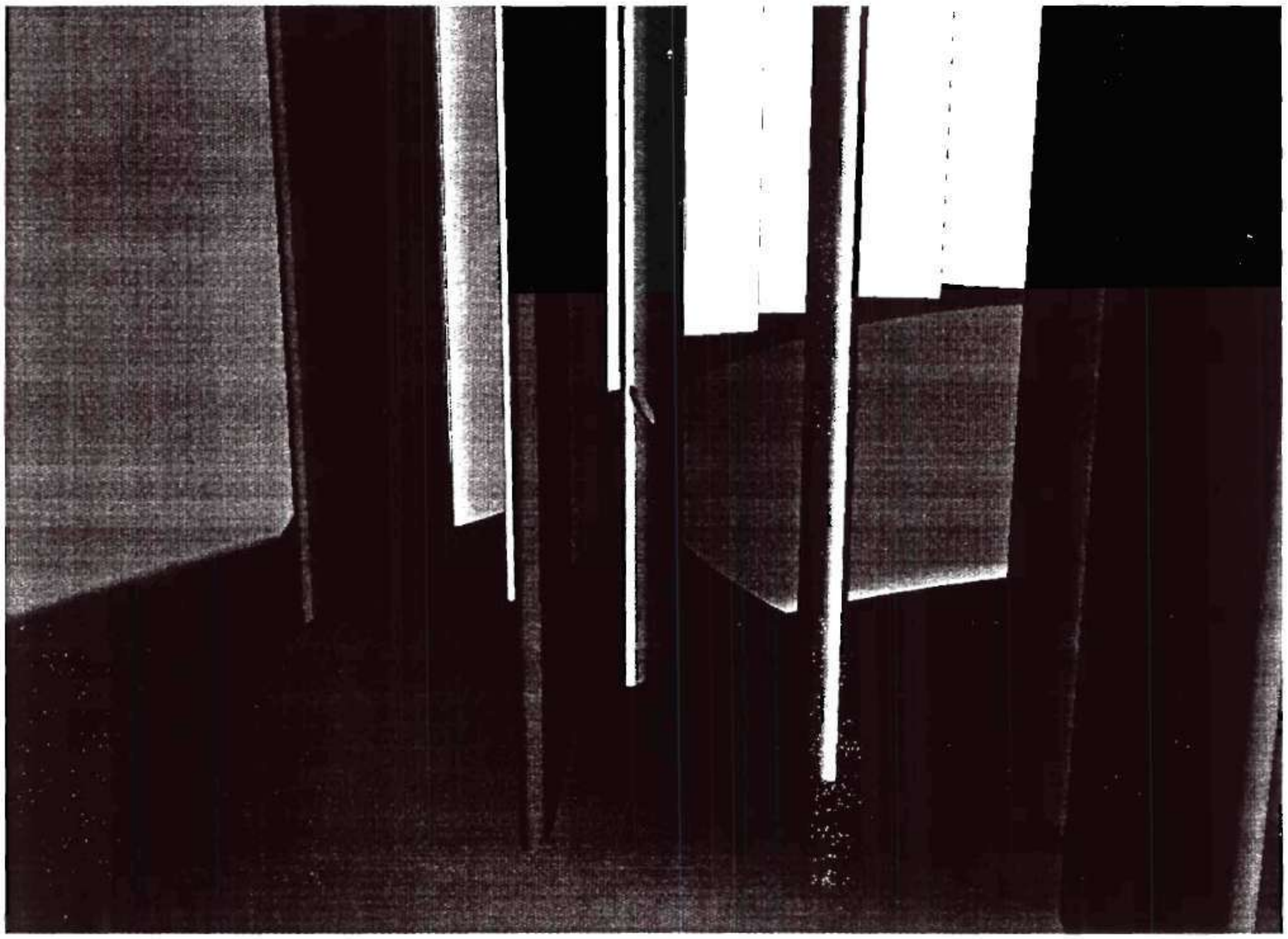


Figure 24. View of a fish-like body before impact on a vane

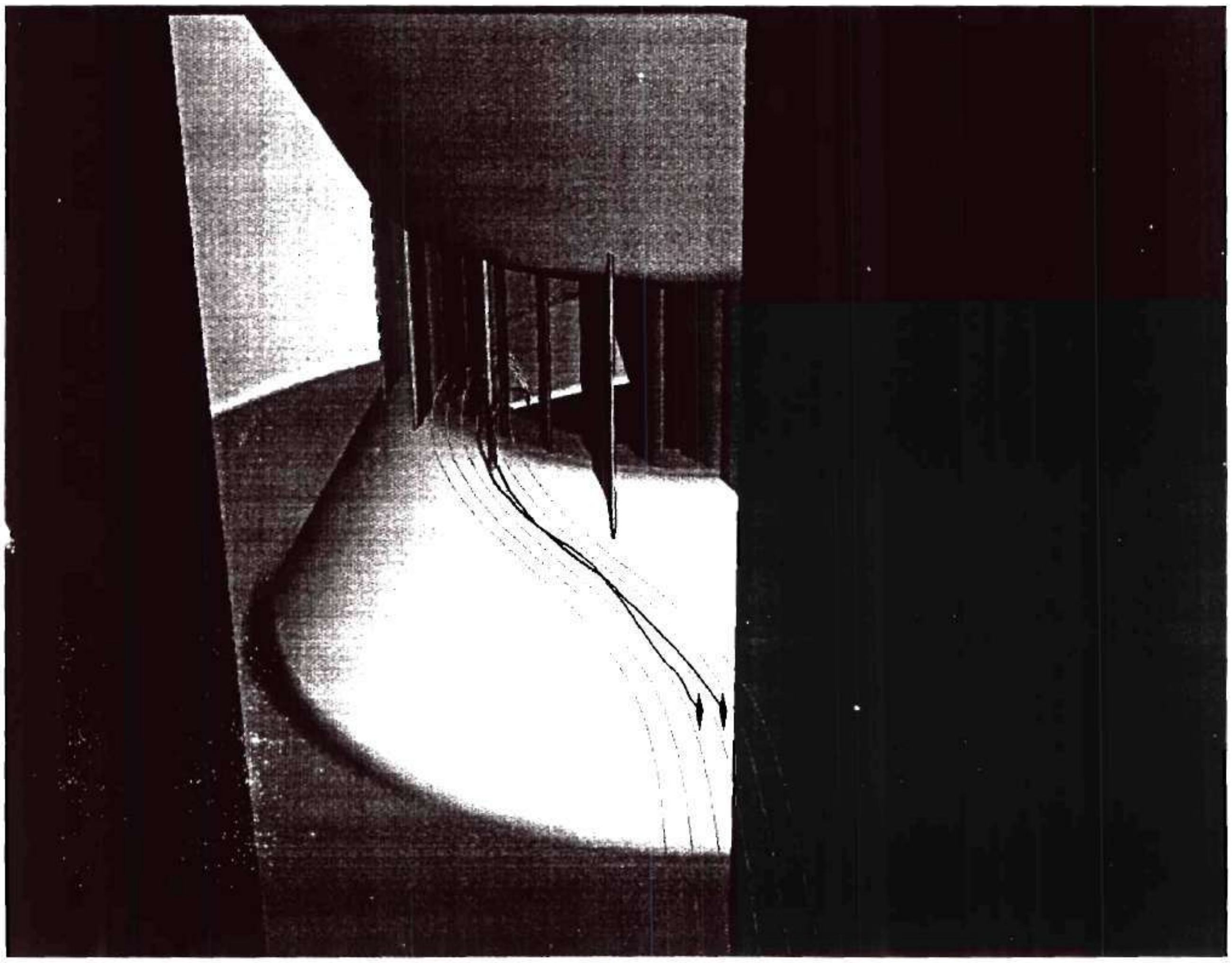


Figure 25. Typical streamlines and fish trajectories through the vanes cascade

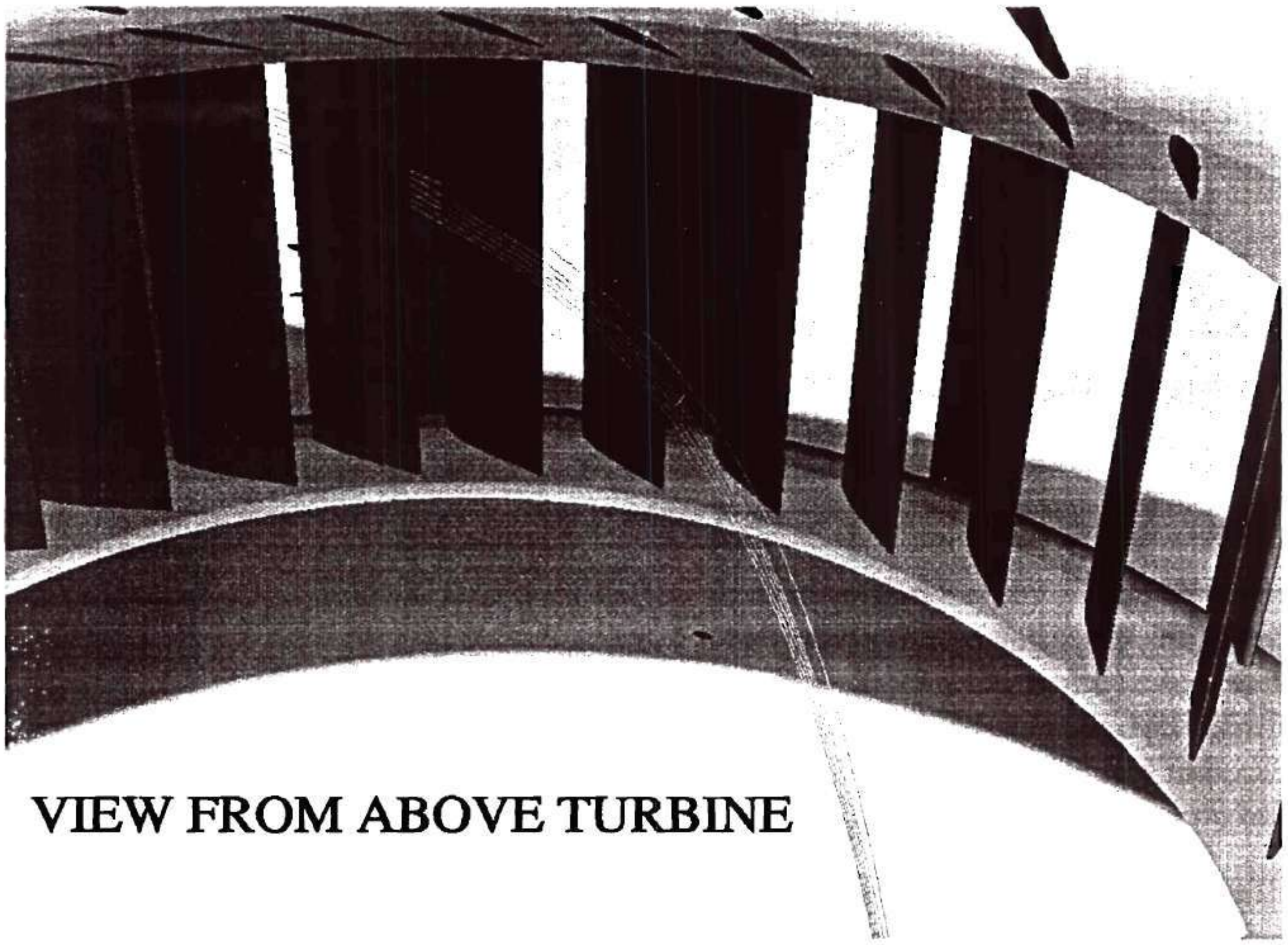


Figure 26. Typical streamlines and fish trajectories through the vanes cascade, view from the turbine shaft

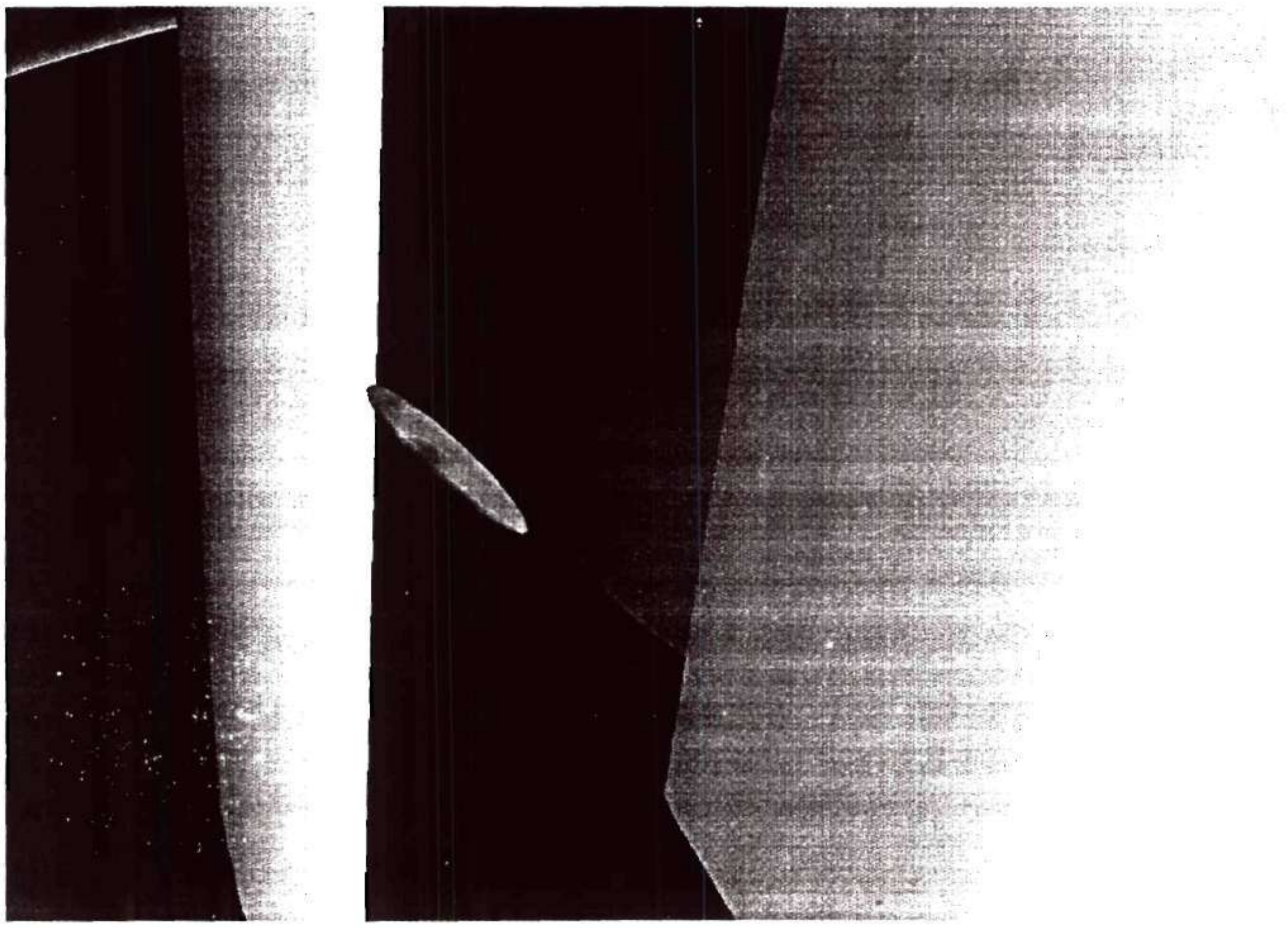


Figure 27. Fish surface pressure used for color-coding fish approaching a vane

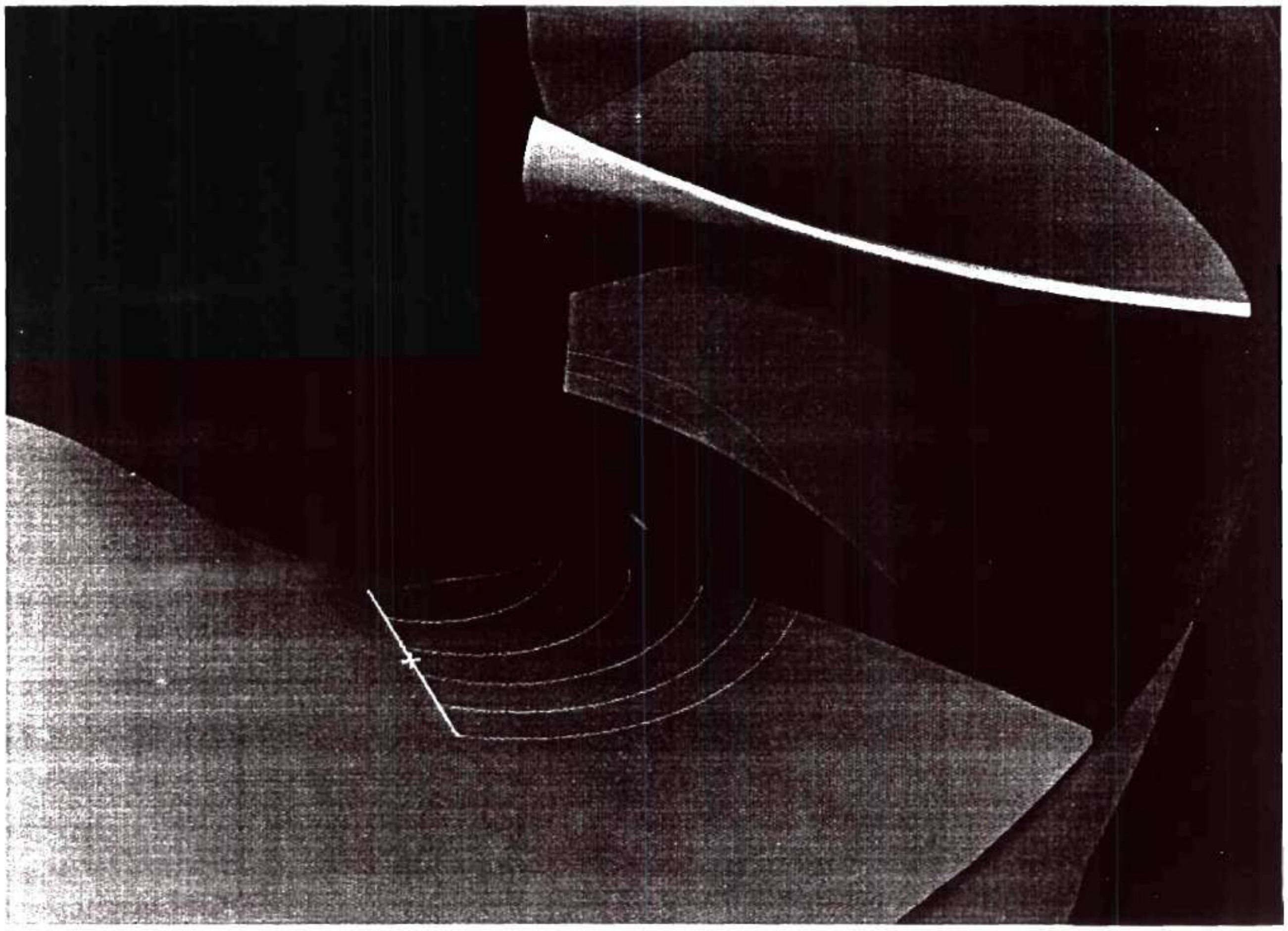


Figure 28. Rake of streamlines and fish trajectory snapshots above a turbine blade

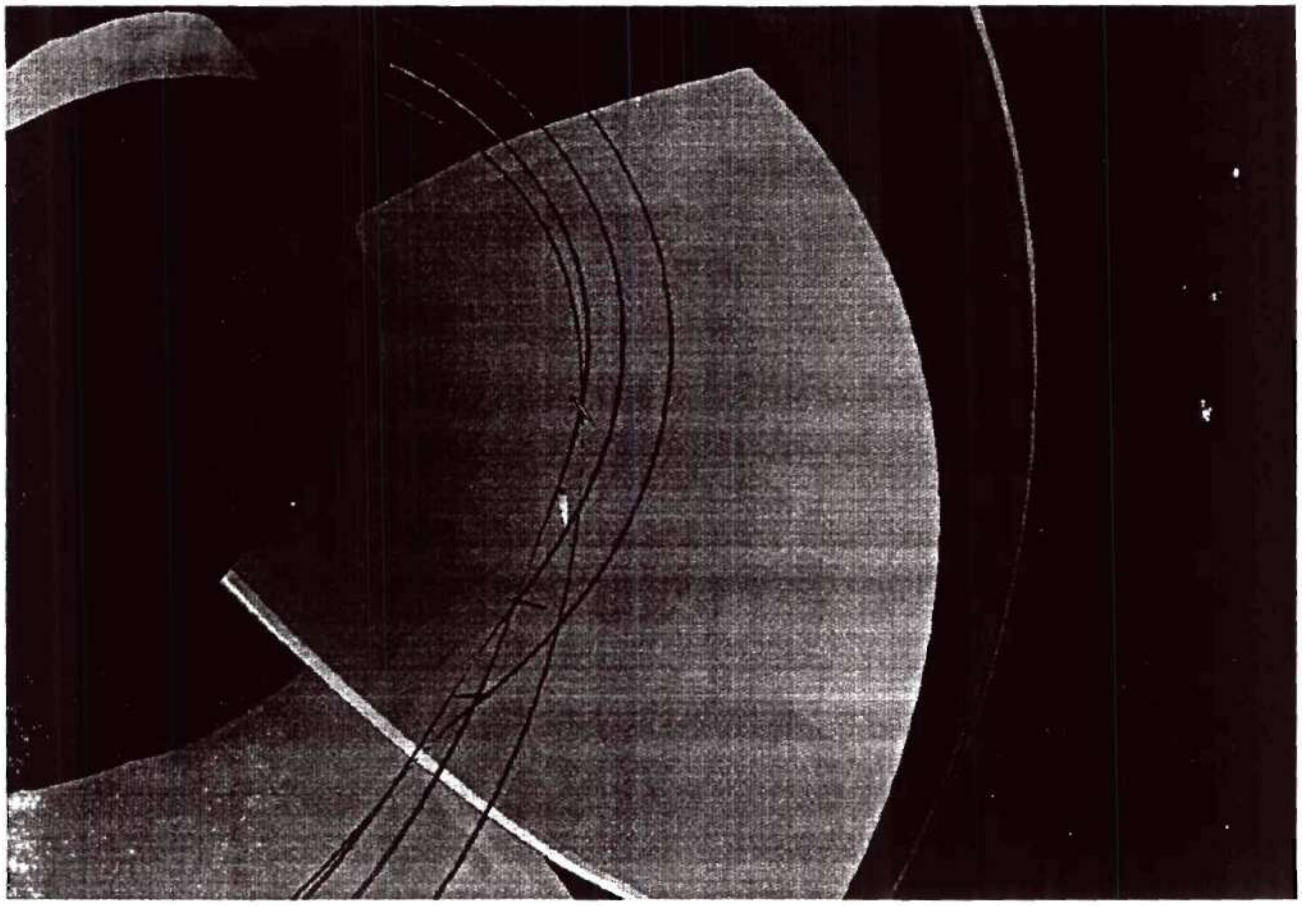


Figure 29. Rake of streamlines and fish trajectory snapshots above a turbine blade

8. Future experiments for model calibration and validation

The brief literature review along with the experience gained during the development and preliminary applications of the VF model, underscore the need for various carefully designed laboratory experiments to supplement our modeling efforts in order to develop a quantitatively accurate design tool. These are summarized below.

1) The empirical correlations for the VF drag and lift coefficients (eqns. 15, 16) used to calculate the force due to the fish-induced viscous flowfield, can be greatly improved using input from suitable experiments. Such experiments can be carried out by placing fish-shaped bodies in a wind-tunnel and measuring the forces exerted on them as function of incidence angle and Reynolds number. This approach should be far more economical than 3-D Navier-Stokes computations around fish-like bodies and could be easily employed to study the aerodynamic properties of a variety of fish species. It should be emphasized that these experiments can be conducted using models that include most real fish characteristics (fins, tail, body roughness, etc.), rather than simplified fish-like bodies. Thus, the resulting data sets will provide very reliable correlations for lift and drag coefficients that can be readily implemented in the model in place of eqns. (15) and (16).

2) Experiments designed to shed light into the response of various fish species to external flow forcing (pressure gradients, vortices, swirl, etc.) are almost entirely lacking from the literature. Yet such information is essential in order to develop a set of behavior rules that could be implemented in conjunction with a neural network to devise a free-will model--such a model can be very easily implemented in the current version of the VF model. Designing and interpreting the results of such experiments will require very close collaboration between CFD modelers and fish biologists.

3) The ultimate objective of the VF model is to provide a tool for estimating fish

mortality. Assuming that we overcome all previously discussed modeling shortcomings, accomplishing this objective further requires input from experiments that are designed to correlate flow-induced loads with injury and mortality. These will provide the necessary threshold loads that can then be incorporated into the model to develop survivability estimates. Rather than trying to isolate various flow effects (e.g. shear vs. turbulence, etc.), as has been done in the past, we propose to adopt a zonal approach. That is, identify a number of potentially harmful to fish flow zones within the powerplant and design experiments that simulate the essential physics of each such zone. The output of such experiments should be detailed mean flow and turbulence statistics measurements along with information regarding individual fish trajectories. The design and execution of these experiments can be greatly facilitated by simultaneous usage of advanced CFD methods.

4) Finally the VF model is ultimately as good as the accuracy of the CFD solutions that supply the virtual flow environments. A significant road block obstructing the application and testing of advanced CFD methods to real-life hydroturbine flows is the lack of comprehensive experiments for model calibration and validation. Such experiments, focusing on obtaining mean flow and turbulence statistics measurements in various subsystems, are also essential for building a powerful predictive tool.

9. References

Alexander R. M., 1993, "Size speed and buoyancy adaptations in aquatic animals", American Zoologist, 30, 189

Cada G. F., Sale M. J., 1993, "Status of fish passage facilities at nonfederal hydropower projects", Fisheries, 18, 7, 4-12a

Cada G. F., Coutant C. C., 1997, "Development of biological criteria for the design of advanced hydropower turbines", Office of Geothermal Technologies, U.S. Dept. of Energy

Chanson H., 1989, "Flow downstream of an aerator – aerator spacing", J. Hydraulic Research, 27, 4, 519-536

Childress S., 1977, "Mechanics of swimming and flying", Courant Institute of Mathematical Sciences, New York University, N.Y.

Costis C. E., Hoang N. T., Telionis D. P., 1989, "Laminar separating flow over a prolate spheroid", J. Aircraft, 86, 810.

Feathers M. G., Knable A. E., 1983, "Effects of depressurization upon largemouth bass", North American Journal of Fisheries Management, 3, 86-90

Fisher R. K., Brown S., Mathur D., 1997, "The importance of the point of operation of a Kaplan turbine on fish survivability", Waterpower '97, Atlanta, GA

Foye R. E., Scott M., 1965, "Effects of pressure on survival of six species of fish", Transactions of the American Fisheries Society, 94, 88-91

Fu T. C., Shekarriz A., Katz J., Huang T. T., 1994, "The flow structure in the lee of an inclined 6:1 prolate spheroid", J. Fluid Mech., 269, 79

Groves A. B., 1972, "Effects of hydraulic shearing actions on juvenile salmon, NW Fisheries Center, National Marine Fisheries Service, WA

Heisey P. G., Mathur D., Euston E. T., 1995, Fish injury and mortality in spillage and turbine passages",proc. Intl. Conf. Hydropower, Waterpower'95, San Francisco, CA

Heisey P. G., Mathur D., Euston E. T., 1996, "Passing fish safety: a closer look at turbine vs. spillway survival", Hydro Review, 15, 2

Hinze, O., 1970, "Turbulence", McGraw Hill Publ.

Hoerner S., 1965, "Fluid-dynamic drag; practical information on aerodynamic drag and hydrodynamic resistance", Midland Park, N.J.

Killgore K. J., Miller A. C., Conley K. C., 1987, "Effects of turbulence on yolk-sac larvae of paddlefish", Transactions of the American Fisheries Society, 116, 670-673

Knapp R. J., Daily J. W., Hammit F. G., 1970, " Cavitation", McGraw-Hill Book Company

Lagler K. F., Bardach J. E., Miller R. R., 1962, "Ichthyology, John Wiley & Sons, Inc. New York

Lin F., Sotiropoulos F., 1997, "Strongly coupled multigrid method for 3-D incompressible flows using near-wall turbulence closures", ASME J. of Fluids Engineering,119, 314-324

Mathur D., Heisey P. G., Euston E. T. Skalski J. R., Hays S., 1996, "Turbine passage survival estimation for chinook salmon smolts (Oncorhynchus tshawytscha) at a large dam on the Columbia River", Canadian J. of Fisheries and Aquatic Sciences, 53, 542-549

Meier H. U., Kreplin H. P., 1980, "Experimental investigation of the boundary layer transition and separation on a body of revolution", Z. Flugwiss. Weltraumforschung, 4(2), 65

Moore A., Scott A., 1988, "Observations of recently emerged sea trout, Salmo Trutta L., fry in a chalk stream, using a low-light underwater camera", J. Fish Biology, 33, 959-960

Morgan R. P., Ulanowicz R. E., Rasin V. J., Noe L. A., Gray G. B., 1976, "Effects of shear on eggs and larvae of striped bass (morone saxatilis) and white perch (morone americana)", Transactions of the American Fisheries Society, 105, 1, 149-154

Muir, J. F., 1959, "Passage of young fish through turbines", Journal of the Power Division, Proceedings of the American Society of Civil Engineers, 85, 1, 23-46

Ransom B. H., Steig T. W. 1994, "Using hydroacoustics to monitor fish in hydropower dams" Lake and Reservoir Management, 9, 1, 163-169

Schoeneman D. E., Pressey R. T., Junge C. O., 1991, "Mortalities of downstream migrant salmon at McNary Dam", Transactions of the Ameican Fisheries Society, 90, 58-72

Shtaf L. G., Pavlov M. A., Skorobotatov M. A., Barekyan A. S., 1983, "The influence of flow turbulence on fish behavior", J. Ichthyology, 23(2), 129

Sotiropoulos F., Ventikos Y., 1998, "Prediction of flow through a 90 deg bend using linear and non-linear two-equation turbulence models", to appear, AIAA Journal

Su W., Tao B., Xu L., 1993, "Three-dimensional separated flow over a prolate spheroid", AIAA Journal, 31, 11, 2176

TASKFLOW Users Manual, AEA, 1998

Traxler S. L., Murphy B. R., Linton T. L., 1993, "Subsediment seismic explosions do not injure caged fishes in a freshwater reservoir", J. Freshwater Ecology, 8, 1, 73-75

Tsvetkov V. I., Pavlov D. S. Nezdoliy V. K. 1972, "Changes in hydrostatic pressure lethal to the young of some freshwater fish", J. of Ichtyology, 12, 307-318

Turnpenny A. W. H., Davis M. H., Flemming J. M., Davies J. K., 1992, "Experimental studies relating to the passge of fish and shrimps through tidal power turbines", Marine and Freshwater Biology Unit, National Power, Fawley, Southhampton, U.K.

USACE (U.S. Army Corps of Engineers), 1991, Passage of small fish through turbines

USACE (U.S. Army Corps of Engineers), 1995, Turbine Survival Passage Workshop

Vaughn R. A., 1995, "John Day dam – underwater video inspection", Waterpower '95, ASCE, New York, NY.

Ventikos Y., Sotiropoulos F., Patel V. C., 1996, "Prediction of turbulent flow through a hydroturbine draft-tube using a near-wall turbulence closure", Proc. XVII IAHR Symp. On Hydraulic Machinery and Cavitation (Cabrera, Espert, Martinez, Eds.), I, 140-149

Voith Hydro, Inc. Report No:2677-0141, 1997, "Development of environmentally advanced hydropower turbine system design concepts", prepared for the U.S. Department of Energy

Webb P. W., Weihs D., 1994, "Hydrostatic stability of fish with swim bladders: not all fish are unstable", Can. J. Zool., 72, 1149

Wu T., Brokaw C., Brennen C., ed., 1974, "Symposium on Swimming and Flying in Nature", California Institute of Technology

Young F. R., 1989, "Cavitation", McGraw-Hill Book Company, 1989

APPENDIX A: The Virtual Fish code - Users Manual

In the sequel, we shall describe the structure and operation of the Fortran code developed. Text that appears under `Courier fonts` corresponds to file names, code constants, variables and subroutines and in general to elements of the actual computer program. The files necessary for a computation are:

- the source Fortran code `virtual.f`
- the include common block file `common-virtual`
- the executable obtained from compiling the source Fortran code `virtual.f`
- the main data file `data-virtual`
- a grid specification file (name defined in main data file `data-virtual`)
- a set of solution specification files (names defined in main data file `data-virtual`)
- a blanking specification file, used to identify the solid wall regions within the multiblock CFD environment (name defined in main data file `data-virtual`)

Description of the code

The result of the research effort described so far is the Fortran computer code `virtual.f`. The code has been tested on three Unix SGI platforms, the Powerchallenge XL, the Origin 2000 and the Octane. Since the algorithm uses standard Fortran instructions, it is expected to be readily portable to any platform, providing adequate resources are present. The main restriction of the method, as far as computer resources necessary, is the main memory: since complex hydraulic machinery components require complex grid topologies to be described accurately, it is commonplace for CFD codes to employ numerous grid blocks, with many nodes each. The need to accommodate such demanding CFD solutions and on the same time be time-efficient, makes the algorithm usable only on medium-high memory sized computers. The hardware and software requirements for a successful execution of the code are:

REQUIREMENTS	Minimum	Suggested
CPU	RISK processor	Last generation RISK processor (Alpha, R10K, Ultra)
MEMORY	256 Mbytes	1024 Mbytes
HARD DISK SPACE	100 Mbytes per CFD solution (excluding the solution files)	200 Mbytes per CFD solution (excluding the solution files)
OPERATING SYSTEM	UNIX	UNIX
FORTRAN COMPILER	ANSI Fortran	Fortran 90

Table 1. System requirements

A compromise between modularity/adaptability and execution speed has been made. More specifically, core parts of the algorithm that are extremely time consuming and are not bound to serious updates in future versions, are quite efficiently but rather obscurely coded. On the other hand, most of the physical modeling part is very easy to adapt and upgrade.

The global variable approach has been used during the construction of the code, meaning that most variables are globally addressable throughout the code. To facilitate this, the use of a single include file containing all the variable definitions has been implemented and call from every subroutine of the code.

A single data file (named `data-virtual`) is used to specify all user supplied data to the code. The structure of this file is described in the sequel. The grid and solution files necessary to run the code correspond to the format of the Voith Hydro TaskFlow solver. An average Fortran programmer can very easily alter the appropriate read statements in the `readfield` subroutine to enable the code to input differently formatted data.

The program is distributed in three forms: a source code file, an executable and a set of subroutines accompanied by a suitably prepared `makefile` file. In this manner, the program can be executed immediately (providing proper UNIX environment settings have been adjusted), it can be re-compiled as a whole, or it can be compiled module-per-module in order serious development is to take place. The latter compilation method is very useful since it cuts down on the overhead of recompiling unaltered program segments.

Summary of the solution algorithm

A detailed description of the solution algorithm is provided herewith.

1. a data file containing instructions for the specific run is read
2. the grid and field data blocks of the CFD solution are read
3. each fish to be simulated is defined geometrically, physically and it is positioned in a predefined initial location at a predefined initial orientation and velocity. Equations (8) to (11) are used to define all necessary auxiliary orientation quantities
4. each node of the fish surface mesh is scanned and the cell and grid block that it is contained is determined, following the technique described in section 4.4.2
5. if the fish node is determined to be on a solid boundary (in the sense described in section 4.5), that node is tagged as impact node and vertical to the wall velocities are computed. An additional force term plus a modified center of rotation are saved not the next steps
6. the values for the velocity components, pressure and turbulence kinetic energy are interpolated from the eight cell vertices to the fish surface node, following equation (21)
7. using the values estimated at the previous step, the forces acting on the fish are defined, as described in section 4.3
8. the equations of motion (6) and (7) are integrated in time according to the scheme presented in 4.4.1 and equation (20).
9. the new fish surface mesh node coordinates are computed from the translations and rotations estimated on the previous step
10. all the quantities of interest are stored, to be used as input for the postprocessor subsequently

The subroutines of the code

The most important subroutines and functions of the FORTRAN program are listed along with a brief description of every subroutine's usage. An attempt has been made to make the program as modular as possible, in order to achieve ease in maintenance, upgrades and troubleshooting.

```
program virtual_fish
```
This main program segment reads the control data file named "data_virtual", calls all the initialization subroutines and loops through the various fish trajectories to be simulated

```
subroutine locate
```
This subroutine searches "cleverly" the cells of all grid blocks in order to pinpoint the position of each individual fish surface node

```
subroutine interpol
```
For every surface node of the fish, this subroutine interpolates the flow variables from the neighbouring grid cell corners, using an inverse distance operator

```
subroutine march
```
This subroutine computes the various force contributions for every node and for the whole fish and integrates the equations of motion in order to propagate the fish in the flow domain

```
integer function icheck
xf1,xijk,xi1jk,xij1k,xijk1,+yf1,yijk,yi1jk,yij1k,yijk1,zf1,
zijk,zi1jk,zij1k,zijk1,acura)
```
This function examines if a fish node is in a particular cell of the grid or not

```
real function volu(x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4)
```
This function computes the volume of a tetrahedron with vertices $(x,y,z)_{1,2,3,4}$

```
subroutine readfield
```
This subroutine reads the grid, velocity, pressure and turbulence multiblock solutions as they are computed by the CFD solver

```
subroutine dimensions
```
This subroutines scales the CFD grid and solution from model to full scale

```
subroutine constants
```
This subroutine sets useful global constants

`subroutine fishinit`
This subroutine sets initial values for all important fish species properties and also establishes the fish surface mesh

`subroutine cdcoef`
This subroutine estimates the drag coefficient for the fish

`subroutine prepare`
This subroutine precomputes main cell volumes for speed

```
subroutine inivol
(xf1,xijk,xi1jk,xij1k,xijk1,
+yf1,yijk,yi1jk,yij1k,yijk1,zf1,zijk,zi1jk,zij1k,zijk1,volare)
```
This is the subroutine where the actual volumes are computed initially for speed.

`subroutine geom`
This subroutine pre-computes the directions of the grid cells, to speed up subsequent force computations.

`subroutine dtcompute`
This routine computes the optimum time step to be used in the integration procedure.

`subroutine impact`
This subroutine estimates wheather the fish has come in contact with the solid wall, and if such an event has occurred, computes the additional forces such an event implies.

`subroutine record`
Subroutine record performs most of the output for the VF model by writing time histories and animation sequences to appropriate files.

`subroutine rotation`
This routine performs the transformation of the flow velocity field from arotating to a stationary frame of reference and vice versa, if such a transformation is necessary.

```
subroutine pres_scl_scroll
subroutine pres_scl_run
subroutine pres_scl_dt
```
These three routines re-dimensionalize the pressure field in the hydraulic component. Only one of these three is called in the beginning of an indivudual run of the VF model.

The most important variables of the code

Description and usage for some of the key variable of the code is given in this section. All dimensional variables are in metric.

`iblock`

is the current CFD grid block number the computations is in

`ix(iblmax),iy(iblmax),iz(iblmax),ibl(imax,jmax,kmax,iblmax)`

hold the i,j,k dimensions of the various grid blocks of the CFD solution

`x(imax,jmax,kmax,iblmax),y(imax,jmax,kmax,iblmax),z(imax,jmax,kmax,iblmax)`

store the grid node coordinates of the various grid blocks of the CFD solution

```
u(imax,jmax,kmax,iblmax),v(imax,jmax,kmax,iblmax),w(imax,jmax,kmax,iblmax),
p(imax,jmax,kmax,iblmax),tke(imax,jmax,kmax,iblmax)
```

store the velocity components, pressure and turbulence kinetic energy of the various grid blocks of the CFD solution

`ixf,jxf`

are the number of fish surface grid cells in the lengthwise and girthwise directions respectively

`xfish(ifmax,jfmax),yfish(ifmax,jfmax),zfish(ifmax,jfmax)`

are the current position of the fish surface nodes

```
ufish(ifmax,jfmax),vfish(ifmax,jfmax),wfish(ifmax,jfmax),pfish(ifmax,jfmax),
tkefish(ifmax,jfmax)
```

are the current velocities, pressure etc. on the fish surface nodes as computed by interpolated from the ambient flow field

`umfish,vmfish,wmfish,mu1fish,vm1fish,wm1fish`

are the previous time step and one before previous time step velocity components of the fish `dt`

is the time step used for the integration of the equations, as it is computed by the algorithm

`acura`

is a variable that controls the maximum allowed relative error in the determination of the location of every fish surface node in the CFD grid topology. This variable should be set to values between 0.5 and 1.0. In case the grid is extremely stretched, skewed and with large aspect ratios, roundoff error might interfere in the accurate determination of fish node locations. In that case it is advised to slightly increase this value, up to 1.5

`ilomem(ifmax,jfmax),imem(ifmax,jfmax),jmem(ifmax,jfmax),kmem(ifmax,jfmax)`

these auxiliary variables hold the previous time step position (in i,j,k mode) of each fish surface node, in order to speed up the location and interpolation procedure

`grfile,velfile,presfile,tkefile`

these character variables hold the names of the files that contain the CFD solution data

`geomscale,velscale,ipref,jpref,kpref,ilopref`

these variables are used to scale the CFD grid and solution from model to full scale and correspond to geometrical ratio, velocity ratio, plus pressure scaling and reference data

`rpm`

is the revolutions per minute of the component. Set to –999 is the component is stationary

`xfinit(100),zfinit(100),yfinit(100)`

are the initial positions of the centers of gravity of the fish to be simulated

`xrinit(100),zrinit(100),yrinit(100)`

are the initial orientations of the fish to be simulated in degrees

`ufinit(100),vfinit(100),wfinit(100)`

are the initial velocities of the fish to be simulated in degrees. Set to –999 if fish are to be injected isokinematically `xlfish,hlfish,wlfish,xfmass,cgfish,cbfish`

correspond to the basic dimensions of the fish, its mass, center of gravity and center of buoyancy. The center of gravity and center of buoyancy are given as fraction of the total length, measuring from the tip of the head. In the current version of the code, the center of buoyancy is fixed, however it is very easy to include a biological law that dictates bladder inflation-deflation and translation of center of buoyancy, following the local pressure for instance.

`gi,denw,viscw`

are constants corresponding to the acceleration of gravity, and the properties of water, density and absolute viscocity

`itypecfd`

specifies the type of hydraulic component to be simulated: value 1 is for the intake-scroll-vanes system, value 2 is for the runner and value 3 is for the draft tube. Depending on the value of this variable, proper assignment of the revolutions per minute and for the pressure-redimensionalization variables have to be set, see discussion in the description of the data file section.

A typical data-virtual data file and the significance of the various data items in it

The data file is structured in such a way that a descriptive line preceeds every actual data line, thus it is more or less self explanatory. All units are metric, angles are in degrees and rotation speed is in revolutions per minute. It should be noted that one pair of lines in the `data-virtual` file correspond to the necessary input for pressure re-dimensionalization. Depending wheather a run for a scroll, a turbine or a draft tube is performed, different structure for these two lines is necessary, since the VF code requires different quantities for each run. The following sample file, as well as the computer file provided, include all three possibilities (leading to an additional 4 lines of data in this file), for reasons of demonstration and completeness. The user should modify this file by removing the two pairs of lines that are redundant and keep just the pair that is pertinent to the run to be performed. The block of lines under discussion is marked in the datafile that follows.

```
Grid file in plot3d format (binary)
plot3d.grd
Velocity file in plot3d format (binary)
velocity
Pressure file in plot3d format (binary)
P
Blanking file for wall definition (ascii)
plot3d.wall
Turbulence kinetic energy file in plot3d format (binary)
TKE
Number of fish trajectories to simulate and their initial coordinates
2                                  ← Number of fish per run
0.0E+00  0.35     -0.16            ← x,y,z initial position of fish No 1
50. 40. 6.                         ← φ,θ,ζ initial angles of orientation of fish No 1
-999. -999. -999.                  ← u,v,w initial velocity of fish No 1
0.0E+00  0.35     -2.49E-02        ← x,y,z initial position of fish No 2
50. 40. 6.                         ← φ,θ,ζ initial angles of orientation of fish No 2
-999. -999. -999.                  ← u,v,w initial velocity of fish No 2
```

```
IxJ fish surface grid
11 11
Length,Width,Height,center of gravity,center of boyancy,density of fish
0.25 .05 .03 .1 .1 1000.
Geometry and Velocity Scale of powerplant (full scale to CFD)
20. 1.714
itypecfd Inlet-Scroll (1)   Runner (2)    Draft tube (3)
1
Href Pcfdinlet Vinlet   hLvsoverH   sigmaPHW  Pvapor PrefProto Zref (scroll)
7.78  35400       0.6096   0.0         1.81292  2450   241800    0.1289
Href HLoverHREF VSQexover2gHref Pvapor PrefProto sigmaP Zref VsqDTE (runner)
7.78 0.02       0.142           2450   241800    0.812  0.1289  0.
Href HLoverHREF VSQexover2gHref Pvapor PrefProto sigmaP Zref    VsqDTE (dt)
7.78 0.02       0.142           2450   241800    0.812  0.1289  0.
Locator accuracy factor (suggested: 0.9)
0.9
Rotating elements rmps (-999. if none). Voith conventions mean positive rpm
sign.
-999.
Write output every n steps, write animation file every n steps
50 1000
Select Tecplot (1), Ensight(2), both(3) or none(0) animation output
1
```

Pressure redimensionalization. Only one of these pairs of lines should be kept in the running version.

The *common-virtual INCLUDE* file

The common block that follows is included in every subroutine of the program and is the primary way of communication between the subroutines. The first line of the common block corresponds to a parameter statement declaring the maximum CFD grid and solution sizes fom the I,j and k directions (`imax, jmax, kmax`) respectively. The constant `iblmax` sets the maximum number of blocks the CFD data may have. It should be noted that these numbers correspond to the **maximum** dimensions, the actual dimensions to be used are defined dynamically in the CFD grid and solution files. However, since the crucial restriction for successfully runing this program is its memory requirements, it is strongly suggested that the current desired values are set in the common block and the executable is regenerated through compilation and linking.

The second line is another parameter statement seting the number of surface grid cells to be generated on the fish body, (`ifmax, jfmax`). Again this corresponds to a maximum number, the actual number is determined in the data file data-virtual.

```
      parameter(imax=25,jmax=89,kmax=73,iblmax=137)
      parameter(ifmax=11,jfmax=11,mxtr=5000)
      common/geom/ iblock,x(imax,jmax,kmax,iblmax),
     +y(imax,jmax,kmax,iblmax),z(imax,jmax,kmax,iblmax),
     +ix(iblmax),iy(iblmax),iz(iblmax),ibl(imax,jmax,kmax,iblmax),
     +wall(imax,jmax,kmax,iblmax)
      common /var/ u(imax,jmax,kmax,iblmax),
     +v(imax,jmax,kmax,iblmax),w(imax,jmax,kmax,iblmax),
     +p(imax,jmax,kmax,iblmax),tke(25,89,73,137)
      common /fish/ xfish(ifmax,jfmax),yfish(ifmax,jfmax),
     +zfish(ifmax,jfmax),ufish(ifmax,jfmax),vfish(ifmax,jfmax),
     +wfish(ifmax,jfmax),pfish(ifmax,jfmax),tkefish(ifmax,jfmax),
     +umfish,vmfish,wmfish,um1fish,vm1fish,wm1fish,um2fish,vm2fish,
     +wm2fish,cdvforce,vvfish,xix,xiy,xiz,ixf,jxf,poldfish(ifmax,jfmax)
      common/rot/omegax,omegay,omegaz,omegaxm1,omegaym1,omegazm1,tpx,
     +tpy,tpz,thetax,thetay,thetaz,theta
      common/int/dt,xp,yp,zp,istep,acura,ilomem(ifmax,jfmax),
     +imem(ifmax,jfmax),jmem(ifmax,jfmax),kmem(ifmax,jfmax),itrajend,
     +xpm1,ypm1,zpm1,xpm2,ypm2,zpm2
      common/files/ grfile,velfile,presfile,tkefile,wallfile
      common/scale/geomscale,velscale,ipref,jpref,kpref,
     +ilopref,rpm,irunner
      common/initfishp/ xfinit(mxtr),zfinit(mxtr),yfinit(mxtr)
      common/initfishr/ xrinit(mxtr),zrinit(mxtr),yrinit(mxtr)
      common/initfishu/ ufinit(mxtr),vfinit(mxtr),wfinit(mxtr)
      common/species/ xlfish,hlfish,wlfish,xfmass,cgfish,cbfish,surf,
     +denfish
      common/cnstnts/gi,denw,viscw,pi
      common/bounce/ fxbounce,fybounce,fzbounce
      common/general/notraj,ifish,jfish,n,ievery,time,ieveryanim,
     +iensight
      common/bladder/pmean,umean,vmean,wmean
      common/presRscale/Href,HLoverHREF,VSQexover2gHref,Pvapor,
     +PrefProto,sigmaP,Zref,VsqDTE,itypecfd
      common/presIscale/Pcfdinlet,Vinlet,hLvsoverH,sigmaPHW
c
      character*30,grfile,velfile,presfile,tkefile,wallfile
```

The listing of the code

A sample listing of the FORTRAN code is presented herewith. Since the actual program is constantly undergoing upgrades and enhancements, this should be viewed as a general guideline, small to moderate changes might be present when compared with newer versions to be released.

```
      program virtual_fish
c
c        *****************************************
c        *                                       *
c        * Virtual Fish Program                  *
c        * Yiannis Ventikos & Fotis Sotiropoulos *
c        * CEE, Georgia Tech 1998                *
c        *                                       *
c        *                                       *
c        * Version 2.5                           *
c        *                                       *
c        *                                       *
c        *****************************************
c
c This main program segment reads the control data file
c named "data_virtual", calls all the initialization
c subroutines and loops through the various fish trajectories
c to be simulated
c
      include'common_virtual'
      character*13 ffile1,ffile2
      character*1 zzz
c
c Open and read data file
      write(*,*) 'Starting simulation'
      open(1,file='data_virtual')
      read(1,200) zzz
      read(1,100) grfile
      read(1,200) zzz
      read(1,100) velfile
      read(1,200) zzz
      read(1,100) presfile
      read(1,200) zzz
      read(1,100) tkefile
      read(1,200) zzz
      read(1,100) wallfile
      read(1,200) zzz
      read(1,*) notraj
      do i=1,notraj
      read(1,*) xfinit(i),yfinit(i),zfinit(i)
      read(1,*) xrinit(i),yrinit(i),zrinit(i)
      read(1,*) ufinit(i),vfinit(i),wfinit(i)
      enddo
      read(1,200) zzz
      read(1,*) ixf,jxf
      read(1,200) zzz
      read(1,*) xlfish,hlfish,wlfish,cgfish,cbfish,denfish
      read(1,200) zzz
      read(1,*) geomscale,velscale
      read(1,200) zzz
      read(1,*) itypecfd
      if (itypecfd.eq.1) then
c read for intake
      read(1,200) zzz
      read(1,*)Href,Pcfdinlet,Vinlet,hLvsoverH,sigmaPHW,
     +Pvapor,PrefProto,Zref
      endif
      if (itypecfd.eq.2) then
c read for runner
      read(1,200) zzz
      read(1,*)Href,HLoverHREF,VSQexover2gHref,
     +Pvapor,PrefProto,sigmaP,Zref,VsqDTE
      endif
      if (itypecfd.eq.3) then
c read for draft tube
      read(1,200) zzz
      read(1,*)Href,HLoverHREF,VSQexover2gHref,
     +Pvapor,PrefProto,sigmaP,Zref,VsqDTE
      endif
      read(1,200) zzz
      read(1,*) acura
      read(1,200) zzz
      read(1,*) rpm
      read(1,200) zzz
      read(1,*) ievery,ieveryanim
      read(1,200) zzz
      read(1,*) iensight
  100 format(a30)
  200 format(80a1)
      close(1)
c
c Set various constants
      call constants
c
c
c
c Read CFD solution
      call readfield
c
c Scale geometry and CFD solution to full scale
      call dimensions
c
c Fix up runner rotation if necessary
c
      irunner=0
      if(rpm.ne.-999.) then
c Convert rpm to rad/sec and adjust for relative frame
      rpm=-(2.*pi)*rpm/60.
      irunner=1
      write(*,*) 'Rotational frame of reference'
      call rotation
      endif
c
c
c Compute optimum time step
      call dtcompute
c
c Loop through all requested trajectories
      do 1 n=1,notraj
      write(*,*) 'Computing trajectory no:', n
      time=0.
      istep=0
      itrajend=0
      close(44)
c     close(55)
      write(ffile1,300) 'FISH1-',n,'.plt'
      write(ffile2,300) 'FISH2-',n,'.plt'
  300 format(a6,i3.3,a4)
      open(66,file='IMPACT_DATA')
      open(44,file=ffile1)
      write(44,*) 'TITLE="FISHDATA"'
      write(44,*) 'VARIABLES=X,Y,Z,TIME,P2,SHRT,SHRU,ACCEL'
      write(44,*) 'ZONE T="1", F=point'
c     open(55,file=ffile2)
c     write(55,*) 'TITLE="FISHDATA"'
c     write(55,*) 'VARIABLES=X,Y,Z,TIME,U,V,W,SHEAR'
c     write(55,*) 'ZONE T="1", F=point'
c
c initialize fish geometry, properties and location
      call fishinit
c
 1000 continue
      time=time+dt
c
c write all important quantities to output files
      istep=istep+1
c
      if(itrajend.eq.1) goto 1
c
c Loop through every fish node
      do ifish=1,ixf
      do jfish=1,jxf
c
c Initialize locators variables
      xp=xfish(ifish,jfish)
      yp=yfish(ifish,jfish)
      zp=zfish(ifish,jfish)
c
c Find the cell each fish node is in
      call locate
c
c Check every fish node for impact.
      call impact
      if(itrajend.eq.1) then
      write(66,66) n,umfish*xfmass,vmfish*xfmass,wmfish*xfmass,
     +xfmass*sqrt(umfish*umfish+vmfish*vmfish+wmfish*wmfish)
      call flush(66)
  66  format(i4,4f15.7)
      goto 1
      endif
      if(itrajend.eq.2) then
      write(*,*) 'impact on wall'
      goto 1
      endif
c
c Interpolate surrounding velocities on every
c fish node
      call interpol
      enddo
      enddo
c
      if(mod(istep,ievery).eq.0) then
      write(*,55)'Traj. ',n,', Step, block, i, j, k for fish "tip": '
     +,istep,ilomem(1,1),imem(1,1),jmem(1,1),kmem(1,1)
 55   format(a6,i4,a38,5i6)
      endif
c
c
c Estimate forces and propagate fish
      call march
c
      goto 1000
c
 1    continue
      stop
      end
c
      subroutine locate
      include'common_virtual'
      dimension ip1(6),ip2(6),ip3(6),ip4(6)
      dimension xt(9),yt(9),zt(9)
c
      data ip1
     +/3,7,7,4,2,3/
      data ip2
     +/4,8,9,5,3,5/
      data ip3
     +/6,9,6,6,5,2/
      data ip4
     +/7,4,4,9,6,6/
c
c This subroutine searches "cleverly" the cells of all
c grid blocks in order to pinpoint the position of each
c individual fish surface node
c
c center-ispan to center+ispan search
      ispan=2
c
      if(istep.eq.1) goto 6688
c
      ilo=ilomem(ifish,jfish)
      istart=imem(ifish,jfish)-ispan
      iend=imem(ifish,jfish)+ispan
      jstart=jmem(ifish,jfish)-ispan
      jend=jmem(ifish,jfish)+ispan
      kstart=kmem(ifish,jfish)-ispan
      kend=kmem(ifish,jfish)+ispan
```

```
c
      if(istart.lt.1) istart=1
      if(jstart.lt.1) jstart=1
      if(kstart.lt.1) kstart=1
c
      if(iend.gt.(ix(ilo)-1)) iend=ix(ilo)-1
      if(jend.gt.(iy(ilo)-1)) jend=iy(ilo)-1
      if(kend.gt.(iz(ilo)-1)) kend=iz(ilo)-1
c
      do 2000 i=istart,iend
      do 3000 j=jstart,jend
      do 4000 k=kstart,kend
c
c find cell
c
      xt(1)=xp
      yt(1)=yp
      zt(1)=zp
      xt(2)=x(i,j,k,ilo)
      yt(2)=y(i,j,k,ilo)
      zt(2)=z(i,j,k,ilo)
      xt(3)=x(i,j,k+1,ilo)
      yt(3)=y(i,j,k+1,ilo)
      zt(3)=z(i,j,k+1,ilo)
      xt(4)=x(i,j+1,k+1,ilo)
      yt(4)=y(i,j+1,k+1,ilo)
      zt(4)=z(i,j+1,k,ilo)
      xt(5)=x(i,j+1,k,ilo)
      yt(5)=y(i,j+1,k,ilo)
      zt(5)=z(i,j+1,k,ilo)
      xt(6)=x(i+1,j,k,ilo)
      yt(6)=y(i+1,j,k,ilo)
      zt(6)=z(i+1,j,k,ilo)
      xt(7)=x(i+1,j,k+1,ilo)
      yt(7)=y(i+1,j,k+1,ilo)
      zt(7)=z(i+1,j,k+1,ilo)
      xt(8)=x(i+1,j+1,k+1,ilo)
      yt(8)=y(i+1,j+1,k+1,ilo)
      zt(8)=z(i+1,j+1,k+1,ilo)
      xt(9)=x(i+1,j+1,k,ilo)
      yt(9)=y(i+1,j+1,k,ilo)
      zt(9)=z(i+1,j+1,k,ilo)
c
      do 5 ilat=1,6
      iii=ip1(ilat)
      jjj=ip2(ilat)
      kkk=ip3(ilat)
      lll=ip4(ilat)
c
c
      idecis=
     +icheck(xt(1),xt(iii),xt(jjj),xt(kkk),xt(lll)
     +,       yt(1),yt(iii),yt(jjj),yt(kkk),yt(lll)
     +,       zt(1),zt(iii),zt(jjj),zt(kkk),zt(lll),acura)
      if (idecis.eq.1) then
      ilomem(ifish,jfish)=ilo
      imem(ifish,jfish)=i
      jmem(ifish,jfish)=j
      kmem(ifish,jfish)=k
      return
      endif
    5 continue
 4000 continue
 3000 continue
 2000 continue
c
 6688 continue
c
c If mini-search fails, Rcheck if first fish node has been located and helps
c
c
      if ((jfish.eq.1).and.(ifish.eq.1)) goto 6689

      ilo=ilomem(1,1)
      istart=imem(1,1)-ispan
      iend=imem(1,1)+ispan
      jstart=jmem(1,1)-ispan
      jend=jmem(1,1)+ispan
      kstart=kmem(1,1)-ispan
      kend=kmem(1,1)+ispan
c
      if(istart.lt.1) istart=1
      if(jstart.lt.1) jstart=1
      if(kstart.lt.1) kstart=1
c
      if(iend.gt.(ix(ilo)-1)) iend=ix(ilo)-1
      if(jend.gt.(iy(ilo)-1)) jend=iy(ilo)-1
      if(kend.gt.(iz(ilo)-1)) kend=iz(ilo)-1
c
      do 2002 i=istart,iend
      do 3002 j=jstart,jend
      do 4002 k=kstart,kend
c
c find cell
c
      xt(1)=xp
      yt(1)=yp
      zt(1)=zp
      xt(2)=x(i,j,k,ilo)
      yt(2)=y(i,j,k,ilo)
      zt(2)=z(i,j,k,ilo)
      xt(3)=x(i,j,k+1,ilo)
      yt(3)=y(i,j,k+1,ilo)
      zt(3)=z(i,j,k+1,ilo)
      xt(4)=x(i,j+1,k+1,ilo)
      yt(4)=y(i,j+1,k+1,ilo)
      zt(4)=z(i,j+1,k,ilo)
      xt(5)=x(i,j+1,k,ilo)
      yt(5)=y(i,j+1,k,ilo)
      zt(5)=z(i,j+1,k,ilo)
      xt(6)=x(i+1,j,k,ilo)
      yt(6)=y(i+1,j,k,ilo)
      zt(6)=z(i+1,j,k,ilo)
      xt(7)=x(i+1,j,k+1,ilo)
      yt(7)=y(i+1,j,k+1,ilo)
      zt(7)=z(i+1,j,k+1,ilo)
      xt(8)=x(i+1,j+1,k+1,ilo)
      yt(8)=y(i+1,j+1,k+1,ilo)
      zt(8)=z(i+1,j+1,k+1,ilo)
      xt(9)=x(i+1,j+1,k,ilo)
      yt(9)=y(i+1,j+1,k,ilo)
      zt(9)=z(i+1,j+1,k,ilo)
c
      do 7 ilat=1,6
      iii=ip1(ilat)
      jjj=ip2(ilat)
      kkk=ip3(ilat)
      lll=ip4(ilat)
c
c
      idecis=
     +icheck(xt(1),xt(iii),xt(jjj),xt(kkk),xt(lll)
     +,      yt(1),yt(iii),yt(jjj),yt(kkk),yt(lll)
     +,      zt(1),zt(iii),zt(jjj),zt(kkk),zt(lll),acura)
      if (idecis.eq.1) then
      ilomem(ifish,jfish)=ilo
      imem(ifish,jfish)=i
      jmem(ifish,jfish)=j
      kmem(ifish,jfish)=k
      return
      endif
    7 continue
 4002 continue
 3002 continue
 2002 continue
c
 6689 continue
c
c If mini-search fails, do the global search
c
      do 1000 ilo=1,iblock
      do 2001 i=1,ix(ilo)-1
      do 3001 j=1,iy(ilo)-1
      do 4001 k=1,iz(ilo)-1
c
c find cell
c
      xt(1)=xp
      yt(1)=yp
      zt(1)=zp
      xt(2)=x(i,j,k,ilo)
      yt(2)=y(i,j,k,ilo)
      zt(2)=z(i,j,k,ilo)
      xt(3)=x(i,j,k+1,ilo)
      yt(3)=y(i,j,k+1,ilo)
      zt(3)=z(i,j,k+1,ilo)
      xt(4)=x(i,j+1,k+1,ilo)
      yt(4)=y(i,j+1,k+1,ilo)
      zt(4)=z(i,j+1,k,ilo)
      xt(5)=x(i,j+1,k,ilo)
      yt(5)=y(i,j+1,k,ilo)
      zt(5)=z(i,j+1,k,ilo)
      xt(6)=x(i+1,j,k,ilo)
      yt(6)=y(i+1,j,k,ilo)
      zt(6)=z(i+1,j,k,ilo)
      xt(7)=x(i+1,j,k+1,ilo)
      yt(7)=y(i+1,j,k+1,ilo)
      zt(7)=z(i+1,j,k+1,ilo)
      xt(8)=x(i+1,j+1,k+1,ilo)
      yt(8)=y(i+1,j+1,k+1,ilo)
      zt(8)=z(i+1,j+1,k+1,ilo)
      xt(9)=x(i+1,j+1,k,ilo)
      yt(9)=y(i+1,j+1,k,ilo)
      zt(9)=z(i+1,j+1,k,ilo)
c
      do 6 ilat=1,6
      iii=ip1(ilat)
      jjj=ip2(ilat)
      kkk=ip3(ilat)
      lll=ip4(ilat)
c
c
      idecis=
     +icheck(xt(1),xt(iii),xt(jjj),xt(kkk),xt(lll)
     +,      yt(1),yt(iii),yt(jjj),yt(kkk),yt(lll)
     +,      zt(1),zt(iii),zt(jjj),zt(kkk),zt(lll),acura)
      if (idecis.eq.1) then
      ilomem(ifish,jfish)=ilo
      imem(ifish,jfish)=i
      jmem(ifish,jfish)=j
      kmem(ifish,jfish)=k
      goto 8000
      endif
    6 continue
 4001 continue
 3001 continue
 2001 continue
 1000 continue
c
c This means (hopefully!) exit of point and end of trajectory
c or bounce back
      itrajend=1
      write(*,*) 'Locate itrajend=1'
 8000 continue
      return
      end
c
c
      subroutine interpol
      include'common_virtual'
c
c For every surface node of the fish, this subroutine
c interpolates the flow variables from the neighbouring
c grid cell corners, using an inverse distance operator
c
      i=imem(ifish,jfish)
      j=jmem(ifish,jfish)
      k=kmem(ifish,jfish)
      ilo=ilomem(ifish,jfish)
c
      dd1=sqrt((x(i,j,k,ilo)-xp)**2 + (y(i,j,k,ilo)-yp)**2 +
     +(z(i,j,k,ilo)-zp)**2)
```

```
       dd2=sqrt((x(i,j+1,k,ilo)-xp)**2 + (y(i,j+1,k,ilo)-yp)**2 +
     +(z(i,j+1,k,ilo)-zp)**2)
       dd3=sqrt((x(i,j+1,k+1,ilo)-xp)**2 + (y(i,j+1,k+1,ilo)-yp)**2 +
     +(z(i,j+1,k+1,ilo)-zp)**2)
       dd4=sqrt((x(i,j,k+1,ilo)-xp)**2 + (y(i,j,k+1,ilo)-yp)**2 +
     +(z(i,j,k+1,ilo)-zp)**2)
       dd5=sqrt((x(i+1,j,k,ilo)-xp)**2 + (y(i+1,j,k,ilo)-yp)**2 +
     +(z(i+1,j,k,ilo)-zp)**2)
       dd6=sqrt((x(i+1,j+1,k,ilo)-xp)**2 + (y(i+1,j+1,k,ilo)-yp)**2 +
     +(z(i+1,j+1,k,ilo)-zp)**2)
       dd7=sqrt((x(i+1,j+1,k+1,ilo)-xp)**2 + (y(i+1,j+1,k+1,ilo)-yp)**2 +
     +(z(i+1,j+1,k+1,ilo)-zp)**2)
       dd8=sqrt((x(i+1,j,k+1,ilo)-xp)**2 + (y(i+1,j,k+1,ilo)-yp)**2 +
     +(z(i+1,j,k+1,ilo)-zp)**2)
c
       uijk=u(i,j,k,ilo)
       uij1k=u(i,j+1,k,ilo)
       uij1k1=u(i,j+1,k+1,ilo)
       uijk1=u(i,j,k+1,ilo)
       ui1jk=u(i+1,j,k,ilo)
       ui1j1k=u(i+1,j+1,k,ilo)
       ui1j1k1=u(i+1,j+1,k+1,ilo)
       ui1jk1=u(i+1,j,k+1,ilo)
c
       vijk=v(i,j,k,ilo)
       vij1k=v(i,j+1,k,ilo)
       vij1k1=v(i,j+1,k+1,ilo)
       vijk1=v(i,j,k+1,ilo)
       vi1jk=v(i+1,j,k,ilo)
       vi1j1k=v(i+1,j+1,k,ilo)
       vi1j1k1=v(i+1,j+1,k+1,ilo)
       vi1jk1=v(i+1,j,k+1,ilo)
c
       wijk=w(i,j,k,ilo)
       wij1k=w(i,j+1,k,ilo)
       wij1k1=w(i,j+1,k+1,ilo)
       wijk1=w(i,j,k+1,ilo)
       wi1jk=w(i+1,j,k,ilo)
       wi1j1k=w(i+1,j+1,k,ilo)
       wi1j1k1=w(i+1,j+1,k+1,ilo)
       wi1jk1=w(i+1,j,k+1,ilo)
c
       pijk=p(i,j,k,ilo)
       pij1k=p(i,j+1,k,ilo)
       pij1k1=p(i,j+1,k+1,ilo)
       pijk1=p(i,j,k+1,ilo)
       pi1jk=p(i+1,j,k,ilo)
       pi1j1k=p(i+1,j+1,k,ilo)
       pi1j1k1=p(i+1,j+1,k+1,ilo)
       pi1jk1=p(i+1,j,k+1,ilo)
c
c      tkeijk=tke(i,j,k,ilo)
c      tkeij1k=tke(i,j+1,k,ilo)
c      tkeij1k1=tke(i,j+1,k+1,ilo)
c      tkeijk1=tke(i,j,k+1,ilo)
c      tkei1jk=tke(i+1,j,k,ilo)
c      tkei1j1k=tke(i+1,j+1,k,ilo)
c      tkei1j1k1=tke(i+1,j+1,k+1,ilo)
```

```
c       tkei1jk1=tke(i+1,j,k+1,ilo)

        if(dd1.lt.0.000000001) dd1=.000000001
        if(dd2.lt.0.000000001) dd2=.000000001
        if(dd3.lt.0.000000001) dd3=.000000001
        if(dd4.lt.0.000000001) dd4=.000000001
        if(dd5.lt.0.000000001) dd5=.000000001
        if(dd6.lt.0.000000001) dd6=.000000001
        if(dd7.lt.0.000000001) dd7=.000000001
        if(dd8.lt.0.000000001) dd8=.000000001

        ekth=-3.5
        dd1=dd1**(ekth)
        dd2=dd2**(ekth)
        dd3=dd3**(ekth)
        dd4=dd4**(ekth)
        dd5=dd5**(ekth)
        dd6=dd6**(ekth)
        dd7=dd7**(ekth)
        dd8=dd8**(ekth)
        dtot=dd1+dd2+dd3+dd4+dd5+dd6+dd7+dd8
        ufish(ifish,jfish)=(dd1*uijk+dd2*uij1k+dd3*uij1k1+dd4*uijk1+
       +dd5*ui1jk+dd6*ui1j1k+dd7*ui1j1k1+dd8*ui1jk1)/dtot
        vfish(ifish,jfish)=(dd1*vijk+dd2*vij1k+dd3*vij1k1+dd4*vijk1+
       +dd5*vi1jk+dd6*vi1j1k+dd7*vi1j1k1+dd8*vi1jk1)/dtot
        wfish(ifish,jfish)=(dd1*wijk+dd2*wij1k+dd3*wij1k1+dd4*wijk1+
       +dd5*wi1jk+dd6*wi1j1k+dd7*wi1j1k1+dd8*wi1jk1)/dtot
        pfish(ifish,jfish)=(dd1*pijk+dd2*pij1k+dd3*pij1k1+dd4*pijk1+
       +dd5*pi1jk+dd6*pi1j1k+dd7*pi1j1k1+dd8*pi1jk1)/dtot
c       tkefish(ifish,jfish)=(dd1*tkeijk+dd2*tkeij1k+dd3*tkeij1k1+dd4*tkeijk1+
c      +dd5*tkei1jk+dd6*tkei1j1k+dd7*tkei1j1k1+dd8*tkei1jk1)/dtot
c
        return
        end
c
c
        subroutine march
        include'common_virtual'
        dimension xxxx(ifmax,jfmax),yyyy(ifmax,jfmax),zzzz(ifmax,jfmax)
c
c This subroutine computes the various force contributions
c for every node and for the whole fish and integrates the
c equations of motion in order to propagate the fish in the
c flow domain
c
c compute average velocity field influencing the fish
        umean=0.
        vmean=0.
        wmean=0.
        pmean=0.
        itotnod=ixf*jxf
        do i=1,ixf
        do j=1,jxf
        umean= umean+ufish(i,j)
        vmean= vmean+vfish(i,j)
        wmean= wmean+wfish(i,j)
        pmean= pmean+pfish(i,j)
        enddo
```

```
      enddo
      umean=umean/float(itotnod)
      vmean=vmean/float(itotnod)
      wmean=wmean/float(itotnod)
      pmean=pmean/float(itotnod)
c
c necessary initializations if new fish is to be "injected"
      if (istep.eq.1) then
      umfish=umean
      vmfish=vmean
      wmfish=wmean
c
      um1fish=umean
      um2fish=umean
      vm1fish=vmean
      vm2fish=vmean
      wm1fish=wmean
      wm2fish=wmean
      do i=1,ixf
      do j=1,jxf
      poldfish(i,j)=pmean
      enddo
      enddo
      if (ufinit(n).gt.(-998.)) then
      umfish=ufinit(n)
      um1fish=ufinit(n)
      um2fish=ufinit(n)
      vmfish=vfinit(n)
      vm1fish=vfinit(n)
      vm2fish=vfinit(n)
      wmfish=wfinit(n)
      wm1fish=wfinit(n)
      wm2fish=wfinit(n)
      endif
      endif

c
c compute relative velocities
      urel=umfish-umean
      vrel=vmfish-vmean
      wrel=wmfish-wmean
      velrel=sqrt(urel*urel+vrel*vrel+wrel*wrel)
      call cdcoef(velrel)
      pold=pmean
c
c effective mass
      acoef=0.5
      xmass=xfmass+acoef*vvfish*denw
c
c viscous components
      fv=-cdvforce*velrel*velrel*surf*0.5*denw
      if(velrel.eq.0.) then
      fxv=0.
      fyv=0.
      fzv=0.
      else
      fxv=fv*urel/velrel
      fyv=fv*vrel/velrel
      fzv=fv*wrel/velrel
      endif
c
c pressure forces
c
      xpress=0.
      ypress=0.
      zpress=0.
      tpz=0.
      tpx=0.
      tpy=0.
      do i=1,ixf-1
      do j=1,jxf-1
c compute
c
      xm1=0.5*(xfish(i,j)+xfish(i+1,j))
      xm2=0.5*(xfish(i,j+1)+xfish(i+1,j+1))
      vect1x=xm2-xm1
      xm3=0.5*(xfish(i,j)+xfish(i,j+1))
      xm4=0.5*(xfish(i+1,j)+xfish(i+1,j+1))
      vect2x=xm4-xm3
c
      ym1=0.5*(yfish(i,j)+yfish(i+1,j))
      ym2=0.5*(yfish(i,j+1)+yfish(i+1,j+1))
      vect1y=ym2-ym1
      ym3=0.5*(yfish(i,j)+yfish(i,j+1))
      ym4=0.5*(yfish(i+1,j)+yfish(i+1,j+1))
      vect2y=ym4-ym3
c
      zm1=0.5*(zfish(i,j)+zfish(i+1,j))
      zm2=0.5*(zfish(i,j+1)+zfish(i+1,j+1))
      vect1z=zm2-zm1
      zm3=0.5*(zfish(i,j)+zfish(i,j+1))
      zm4=0.5*(zfish(i+1,j)+zfish(i+1,j+1))
      vect2z=zm4-zm3
      ds1=sqrt(vect1x*vect1x+vect1y*vect1y+vect1z*vect1z)
      ds2=sqrt(vect2x*vect2x+vect2y*vect2y+vect2z*vect2z)
      da=ds1*ds2
c
      dpa=0.25*(pfish(i,j)+pfish(i,j+1)+pfish(i+1,j+1)+pfish(i+1,j))
      dfpressa=da*dpa
c
c decompose
c
      xnorm=vect1y*vect2z-vect1z*vect2y
      ynorm=vect1z*vect2x-vect1x*vect2z
      znorm=vect1x*vect2y-vect1y*vect2x
      snorm=sqrt(xnorm*xnorm+ynorm*ynorm+znorm*znorm)
      xpres1=dfpressa*xnorm/snorm
      ypres1=dfpressa*ynorm/snorm
      zpres1=dfpressa*znorm/snorm
      xpress=xpress+xpres1
      ypress=ypress+ypres1
      zpress=zpress+zpres1
c
c Torque contributions
c
      darm=sqrt((xfish(i,j)-xpm1)**2+(yfish(i,j)-ypm1)**2+
     +(zfish(i,j)-zpm1)**2)
      tpz=tpz+darm*xpres1
      tpz=tpz+darm*ypres1
      tpy=tpy+darm*xpres1
      tpy=tpy+darm*zpres1
      tpx=tpx+darm*ypres1
      tpx=tpx+darm*zpres1
c
      enddo
      enddo
  888 continue
      xpress=-xpress
      ypress=-ypress
      zpress=-zpress
c
c buoyancy
      dendif=denw-denfish
      bforce=dendif*vvfish*gi
c
c total force
      fx=fxv+xpress+fxbounce
      fy=fyv+ypress+fybounce
      fz=fzv+zpress+bforce+fzbounce
      fx=fxv+fxbounce
      fy=fyv+fybounce
      fz=fzv+bforce+fzbounce
c
c integrate twice for translatory
c velocities
      umfish=
     +(4.*um1fish-um2fish+(2.*dt*fx/xmass))/3.
      vmfish=
     +(4.*vm1fish-vm2fish+(2.*dt*fy/xmass))/3.
      wmfish=
     +(4.*wm1fish-wm2fish+(2.*dt*fz/xmass))/3.
c add rotation of runner to fish speed if necessary
      if (irunner.eq.1) then
      umfish=umfish+((-1.)*ypm1*rpm)
      vmfish=vmfish+((-1.)*xpm1*(-1.)*rpm)
      endif
c positions
      xp=(4.*xpm1-xpm2+(2.*dt*umfish))/3.
      yp=(4.*ypm1-ypm2+(2.*dt*vmfish))/3.
      zp=(4.*zpm1-zpm2+(2.*dt*wmfish))/3.
c remove rotation of runner until next step...
      if (irunner.eq.1) then
      umfish=umfish-((-1.)*ypm1*rpm)
      vmfish=vmfish-((-1.)*xpm1*(-1.)*rpm)
      endif
c
c
c integrate for rotational
      omegax=omegaxm1+dt*(tpx+omegay*omegaz*
     *(xiy-xiz))/xix
      thetax=thetax+omegax*dt
      omegay=omegaym1+dt*(tpy+omegax*omegaz*
     *(xiz-xix))/xiy
      thetay=thetay+omegay*dt
      omegaz=omegazm1+dt*(tpz+omegax*omegay*
     *(xix-xiy))/xiz
      thetaz=thetaz+omegaz*dt
c
c fix rotations and orientations
c
c     update position of fish
      do i=1,ixf
      do j=1,jxf
      xfish(i,j)=xfish(i,j)+(xp-xpm1)
      yfish(i,j)=yfish(i,j)+(yp-ypm1)
      zfish(i,j)=zfish(i,j)+(zp-zpm1)
      enddo
      enddo
c
c rotate fish
c
c localize
      xpce=0.
      ypce=0.
      zpce=0.
      do i=1,ixf
      do j=1,jxf
      xpce=xpce+xfish(i,j)
      ypce=ypce+yfish(i,j)
      zpce=zpce+zfish(i,j)
      enddo
      enddo
      xpce=xpce/(float(ixf*jxf))
      ypce=ypce/(float(ixf*jxf))
      zpce=zpce/(float(ixf*jxf))
      do i=1,ixf
      do j=1,jxf
      xxxx(i,j)=xfish(i,j)-xpce
      yyyy(i,j)=yfish(i,j)-ypce
      zzzz(i,j)=zfish(i,j)-zpce
      enddo
      enddo
c
      do ii=1,ixf
      do jj=1,jxf
      xxx=xxxx(ii,jj)*cos(thetaz)+yyyy(ii,jj)*sin(thetaz)
      yyy=-xxxx(ii,jj)*sin(thetaz)+yyyy(ii,jj)*cos(thetaz)
      xxxx(ii,jj)=xxx
      yyyy(ii,jj)=yyy
      enddo
      enddo
      do ii=1,ixf
      do jj=1,jxf
      xxx=xxxx(ii,jj)*cos(thetay)+zzzz(ii,jj)*sin(thetay)
      zzz=-xxxx(ii,jj)*sin(thetay)+zzzz(ii,jj)*cos(thetay)
      xxxx(ii,jj)=xxx
      zzzz(ii,jj)=zzz
      enddo
      enddo
      do ii=1,ixf
      do jj=1,jxf
      yyy=yyyy(ii,jj)*cos(thetax)+zzzz(ii,jj)*sin(thetax)
      zzz=-yyyy(ii,jj)*sin(thetax)+zzzz(ii,jj)*cos(thetax)
      yyyy(ii,jj)=yyy
      zzzz(ii,jj)=zzz
      enddo
      enddo
c
c substitute
      do i=1,ixf
      do j=1,jxf
      xfish(i,j)=xxxx(i,j)+xpce
      yfish(i,j)=yyyy(i,j)+ypce
      zfish(i,j)=zzzz(i,j)+zpce
      enddo
      enddo
      omegax=0.
      omegay=0.
      omegaz=0.
      thetax=0.
      thetay=0.
      thetaz=0.
      theta=amax1(thetax,thety,thetaz)
c
c backsubstitute
      xpm2=xpm1
      xpm1=xp
      ypm2=ypm1
      ypm1=yp
      zpm2=zpm1
      zpm1=zp
      um2fish=um1fish
      um1fish=umfish
      vm2fish=vm1fish
      vm1fish=vmfish
      wm2fish=wm1fish
      wm1fish=wmfish
      omegaxm1=omegax
      omegaym1=omegay
      omegazm1=omegaz
c
c write all important quantities to output files
      call record
c
c Update pressure on fish
      do i=1,ixf
      do j=1,jxf
      poldfish(i,j)=pfish(i,j)
      enddo
      enddo
      return
      end
c
      integer function icheck(xf1,xijk,xi1jk,xij1k,xijk1,
     +yf1,yijk,yi1jk,yij1k,yijk1,zf1,zijk,zi1jk,zij1k,zijk1,acura)
c
c This function examines if a fish node is in
c a particular cell of the grid or not
c
      icheck=0
```

```
c
c volume of main tetrahedron (this is precomputed)
c
      vol1f=volu(xijk,xi1jk,xij1k,xijk1,
     +           yijk,yi1jk,yij1k,yijk1,
     +           zijk,zi1jk,zij1k,zijk1)
c
c subvolumes inside tetrahedron 1
c
c subvolume 1
c
      vol11=volu(xf1,xi1jk,xij1k,xijk1,
     +           yf1,yi1jk,yij1k,yijk1,
     +           zf1,zi1jk,zij1k,zijk1)
c
c subvolume 2
c
      vol12=volu(xf1,xijk,xij1k,xijk1,
     +           yf1,yijk,yij1k,yijk1,
     +           zf1,zijk,zij1k,zijk1)
c
c subvolume 3
c
      vol13=volu(xf1,xi1jk,xijk,xijk1,
     +           yf1,yi1jk,yijk,yijk1,
     +           zf1,zi1jk,zijk,zijk1)
c
c subvolume 4
c
      vol14=volu(xf1,xi1jk,xij1k,xijk,
     +           yf1,yi1jk,yij1k,yijk,
     +           zf1,zi1jk,zij1k,zijk)
c
      vol1c=vol11+vol12+vol13+vol14
      voldif=abs(vol1f-vol1c)/vol1f
      if (voldif.le.acura) then
      icheck=1
      endif
      return
      end
c
c
      real function volu(x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4)
c
c this function computes the volume of a tetrahedron with
c vertices (x,y,z)_1,2,3,4
c
      dx1=x2-x1
      dx2=x3-x1
      dx3=x4-x1
      dy1=y2-y1
      dy2=y3-y1
      dy3=y4-y1
      dz1=z2-z1
      dz2=z3-z1
      dz3=z4-z1
      volu =abs(dx1*dy2*dz3+dy1*dz2*dx3+dz1*dx2*dy3-
     -                 dy1*dx2*dz3-dx1*dz2*dy3-dz1*dy2*dx3)
```

```
c
c Note: The exact formula of the volume of a tetrahedron requires
c multiplication by 1/6, a term that can and is omited (since
c only comparisons of volumes take place) for the sake of performance.
c
      return
      end
c
      subroutine readfield
      include'common_virtual'
c
c This subroutine reads the grid, velocity, pressure and
c turbulence multiblock solutions as they are computed by
c CFD solver
c
c read the grid blocks
      open(10,file=grfile,form='unformatted')
      read(10) iblock
      read(10) (ix(ilo),iy(ilo),iz(ilo),ilo=1,iblock)
      do 101 ilo=1,iblock
      read(10) (((x(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo)),
     +         (((y(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo)),
     +         (((z(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo)),
     +         (((ibl(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
 101  continue
      close(10)
      write(*,*) 'Grid o.k.'
c
c read the velocity blocks
      open(10,file=velfile,form='unformatted')
      read(10) iblock
      read(10) (ix(ilo),iy(ilo),iz(ilo),idum,ilo=1,iblock)
      do 102 ilo=1,iblock
      read(10) (((u(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo)),
     +         (((v(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo)),
     +         (((w(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
 102  continue
      close(10)
      write(*,*) 'Velocity o.k.'
c
c read the pressure blocks
      open(10,file=presfile,form='unformatted')
      read(10) iblock
      read(10) (ix(ilo),iy(ilo),iz(ilo),idum,ilo=1,iblock)
      do 103 ilo=1,iblock
      read(10) (((p(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
 103  continue
      close(10)
      write(*,*) 'Pressure o.k.'
c
c read the turbulence kinetic energy blocks
c     open(10,file=tkefile,form='unformatted')
c     read(10) iblock
c     read(10) (ix(ilo),iy(ilo),iz(ilo),idum,ilo=1,iblock)
c     do 104 ilo=1,iblock
c     read(10)(((tke(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c104  continue
c     close(10)
```

```
c      write(*,*) 'Turbulence data o.k.'
c
c Read the Wall information
       open(10,file=wallfile)
       do ilo=1,iblock
       do i=1,ix(ilo)
       do j=1,jx(ilo)
       do k=1,kx(ilo)
       read(10,*) idum1,idum2,idum3,idum4,wall(idum2,idum3,idum4,idum1)
       enddo
       enddo
       enddo
       enddo
       write(*,*) 'Wall data o.k.'
c
       return
       end
c
       subroutine dtcompute
       include'common_virtual'
c
c This subroutine makes a very conservative estimation of the
c time step to be used for the integration of the equations
c
       dt=10.e30
       do 34 ilo=1,iblock
       do 34 i=2,ix(ilo)-1
       do 34 j=2,iy(ilo)-1
       do 34 k=2,iz(ilo)-1
       xl1=sqrt(
      +((x(i+1,j,k,ilo)-x(i,j,k,ilo))**2)+
      +((y(i+1,j,k,ilo)-y(i,j,k,ilo))**2)+
      +((z(i+1,j,k,ilo)-z(i,j,k,ilo))**2))
       xl2=sqrt(
      +((x(i,j+1,k,ilo)-x(i,j,k,ilo))**2)+
      +((y(i,j+1,k,ilo)-y(i,j,k,ilo))**2)+
      +((z(i,j+1,k,ilo)-z(i,j,k,ilo))**2))
       xl3=sqrt(
      +((x(i,j,k+1,ilo)-x(i,j,k,ilo))**2)+
      +((y(i,j,k+1,ilo)-y(i,j,k,ilo))**2)+
      +((z(i,j,k+1,ilo)-z(i,j,k,ilo))**2))
       xlmin=amin1(xl1,xl2,xl3)
       if(xlmin.lt.0.000001) write(*,*) 'Problem with grid at ',i,j,k
       velmax=sqrt(u(i,j,k,ilo)*u(i,j,k,ilo)+v(i,j,k,ilo)*v(i,j,k,ilo)
      ++w(i,j,k,ilo)*w(i,j,k,ilo))
       dt=amin1(dt,(xlmin/velmax))
    34 continue
c A walk on the wild side: this has spectacular results as far as speedup
and
c has never failed *so far*. If it does in the future, the user is hinted
c to reduce this factor to something more modest.
       dt=4.0*dt
       write(*,*) 'dt computed=',dt
c
       return
       end
c
       subroutine dimensions
      include'common_virtual'
c
c This subroutines scales the CFD grid and solution
c from model to full scale
c Pressure must be scaled and dimensionalized (using sigma)
c BEFORE the grid is scaled
c
c
c scale pressure using sigma values
c
      if(itypecfd.eq.1) then
      call  pres_scl_scroll
      endif
      if(itypecfd.eq.2) then
      call  pres_scl_run
      endif
      if(itypecfd.eq.3) then
      call  pres_scl_dt
      endif
c
c
c scale velocities and grid
c
      do 34 ilo=1,iblock
      do 34 i=1,ix(ilo)
      do 34 j=1,iy(ilo)
      do 34 k=1,iz(ilo)
c
      u(i,j,k,ilo)=u(i,j,k,ilo)*velscale
      v(i,j,k,ilo)=v(i,j,k,ilo)*velscale
      w(i,j,k,ilo)=w(i,j,k,ilo)*velscale
c     tke(i,j,k,ilo)=tke(i,j,k,ilo)*(velscale**2)
      x(i,j,k,ilo)=x(i,j,k,ilo)*geomscale
      y(i,j,k,ilo)=y(i,j,k,ilo)*geomscale
      z(i,j,k,ilo)=z(i,j,k,ilo)*geomscale
c
 34   continue
c
      return
      end
c
      subroutine constants
      include'common_virtual'
c
c This subroutine sets useful global constants.
c Make sure that the constants you add here, appear also
c in the common INCLUDE file "common_virtual"
c
c Density of water is 1000.00 kg/m^3
      denw=1000.00
c
c Dynamic viscocity of water is 1.12e-3 N*s/m^2
      viscw=1.12e-3
c
c Acceleration of gravity on earth is 9.81 m/s^2
      gi=9.81
c
c Pi is pi
```

```
      pi=3.14159
c
      return
      end
c
      subroutine fishinit
      include'common_virtual'
c
c This subroutine sets initial values for all
c important fish species properties and also
c establishes the fish surface mesh
c
c geometrically "build" fish as a prolate ellipsoid
c

      theta=0.
      dth=pi/float(ixf-1)
      dphi=2.*pi/float(jxf-1)
      do i=1,ixf
      phi=0.
      do j=1,jxf
           xfish(i,j) =   cos(theta)
           yfish(i,j) = sin(theta)*sin(phi)
           zfish(i,j) = sin(theta)*cos(phi)
      phi=phi+dphi
        enddo
      theta=theta+dth
        enddo
      do i=1,ixf
      do j=1,jxf
      xfish(i,j)=0.5*xfish(i,j)*xlfish
      yfish(i,j)=0.5*yfish(i,j)*hlfish
      zfish(i,j)=0.5*zfish(i,j)*wlfish
      enddo
      enddo
c


c     dxlfish=xlfish/float(ixf-1)
c     dpp=2.*pi/float(jxf-1)
c     i=0
c     do 11 ik=0,float(ixf-1)
c     xpp=(-xlfish/2.)+(dxlfish*ik)
c     i=i+1
c     j=0
c     do 11 il=0,float(jxf-1)
c     ypp=dpp*il
c     j=j+1
c     xfish(i,j)=xpp
c     yfish(i,j)=hlfish*cos(ypp)*((xlfish/2.)-abs(xpp))/(xlfish/2.)
c     zfish(i,j)=wlfish*sin(ypp)*((xlfish/2.)-abs(xpp))/(xlfish/2.)
c 11  continue
c
      vvfish=1.33*(hlfish/2.)*(xlfish/2.)*(wlfish/2.)*pi
      xfmass=denfish*vvfish
      surf=xlfish*(wlfish+hlfish)*2.
c
c initial rotations
```

```
c
      xrinit(n)=pi*xrinit(n)/180.
      yrinit(n)=pi*yrinit(n)/180.
      zrinit(n)=pi*zrinit(n)/180.
      thetax=xrinit(n)
      thetay=yrinit(n)
      thetaz=zrinit(n)
      do ii=1,ixf
      do jj=1,jxf
      xxx=xfish(ii,jj)*cos(zrinit(n))+yfish(ii,jj)*sin(zrinit(n))
      yyy=-xfish(ii,jj)*sin(zrinit(n))+yfish(ii,jj)*cos(zrinit(n))
      xfish(ii,jj)=xxx
      yfish(ii,jj)=yyy
      enddo
      enddo
      do ii=1,ixf
      do jj=1,jxf
      xxx=xfish(ii,jj)*cos(yrinit(n))+zfish(ii,jj)*sin(yrinit(n))
      zzz=-xfish(ii,jj)*sin(yrinit(n))+zfish(ii,jj)*cos(yrinit(n))
      xfish(ii,jj)=xxx
      zfish(ii,jj)=zzz
      enddo
      enddo
      do ii=1,ixf
      do jj=1,jxf
      yyy=yfish(ii,jj)*cos(xrinit(n))+zfish(ii,jj)*sin(xrinit(n))
      zzz=-yfish(ii,jj)*sin(xrinit(n))+zfish(ii,jj)*cos(xrinit(n))
      yfish(ii,jj)=yyy
      zfish(ii,jj)=zzz
      enddo
      enddo

c
c place fish in desired initial position
      do ii=1,ixf
      do jj=1,jxf
      xfish(ii,jj)=xfish(ii,jj)+xfinit(n)*geomscale
      yfish(ii,jj)=yfish(ii,jj)+yfinit(n)*geomscale
      zfish(ii,jj)=zfish(ii,jj)+zfinit(n)*geomscale
      enddo
      enddo

c     open(22,file='TEST1.plt')
c     write(22,*) 'TITLE="FISHBODY"'
c     write(22,*) 'VARIABLES=X,Y,Z,P,DUM,DUM,DUM,DUM'
c     write(22,*) 'ZONE T="B", I= ',jxf,',J= ',ixf,' , F=block'
c     write(22,*) (((xfish(i,j)/geomscale),j=1,jxf),i=1,ixf)
c     write(22,*) (((yfish(i,j)/geomscale),j=1,jxf),i=1,ixf)
c     write(22,*) (((zfish(i,j)/geomscale),j=1,jxf),i=1,ixf)
c     write(22,*) ((pfish(i,j),j=1,jxf),i=1,ixf)
c     write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
c     write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
c     write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
c     write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
c     close(22)

c
      open(33,file='Fish_Shape')
      do ii=1,ixf
      do jj=1,jxf
      write(33,*) xfish(ii,jj),yfish(ii,jj),zfish(ii,jj)
      enddo
      enddo
      close(33)
c
c center through locus
      xpm1=0.
      xpm2=0.
      ypm1=0.
      ypm2=0.
      zpm1=0.
      zpm2=0.
      do ii=1,ixf
      do jj=1,jxf
      xpm1=xpm1+xfish(ii,jj)
      ypm1=ypm1+yfish(ii,jj)
      zpm1=zpm1+zfish(ii,jj)
      enddo
      enddo
      xpm1=xpm1/float(ixf*jxf)
      ypm1=ypm1/float(ixf*jxf)
      zpm1=zpm1/float(ixf*jxf)
      xpm2=xpm1
      ypm2=ypm1
      zpm2=zpm1
c
c Set the three moments of inertia
      xix=denfish*vvfish*.2*(hlfish**2 + wlfish**2)
      xiy=denfish*vvfish*.2*(xlfish**2 + hlfish**2)
      xiz=denfish*vvfish*.2*(xlfish**2 + wlfish**2)
c
c Initialize angular velocity
      omegax=0.
      omegay=0.
      omegaz=0.
      omegaxm1=0.
      omegaym1=0.
      omegazm1=0.
c
      return
      end
c
c
c*************************************************************
c
      subroutine cdcoef(velrel)
      include'common_virtual'
c
c This subroutine estimates the drag coefficient for the
c fish
c
c fish reynolds number
      ren=abs(denw*xlfish*velrel/viscw)
      if(ren.lt.0.0000001) then
      cdvforce=0.
      return
      endif
c
c turbulent flat plate
      cdft=0.455/((log(ren)/log(10.))**2.58)
c
c fish body cd
      cdfb=cdft*(1.+(1.5*(xlfish/hlfish)**1.5)
     ++7.*(hlfish/xlfish)**3)
c correction for angle
      notheta=int(theta/(pi/2.))
      theta=theta-float(notheta)*(pi/2.)
      cdcor=0.405*theta**2 +0.0014*theta+1.0
c correction for roughness s to be put here, when that info becomes
available. (typical scales size over xlfish=)
      cdcor=cdcor*1.
c final
      cdvforce=cdfb*cdcor
      if(cdvforce.lt.5.) cdvforce=5.
      return
      end

      subroutine record
c
c This soubroutine handles practically all of the output
c
      include'common_virtual'
      character*19,anims
      character*17,animse,animsp
c
      if (mod(istep,ievery).eq.0) then
      shearmax=-10.e30
      pmaxf=-10.e30
      pminf=10.e30
      dpdtmaxf=-10.e30
      dpdtminf=10.e30
      do i=1,ixf
      do j=1,jxf
      pmaxf=amax1(pmaxf,pfish(i,j))
      pminf=amin1(pminf,pfish(i,j))
      dpdt=(pfish(i,J)-poldfish(i,J))/dt
      dpdtmaxf=amax1(dpdtmaxf,dpdt)
      dpdtminf=amin1(dpdtminf,dpdt)
      difu=abs(umfish-umean)
      difv=abs(vmfish-vmean)
      difw=abs(wmfish-wmean)
      shearmax=amax1(difu,difv,difw)
      sheartot=sqrt(difu*difu+difv*difv+difw*difw)
      enddo
      enddo
      speed1=sqrt(umfish*umfish+vmfish*vmfish+wmfish*wmfish)
      speed2=sqrt(um2fish*um2fish+vm2fish*vm2fish+wm2fish*wm2fish)
      accel=(speed1-speed2)/dt
      dpdt=(pfish(2,2)-poldfish(2,2))/dt
      dpdtmaxf=dpdt
      dpdtminf=dpdt
      pmaxf=pfish(2,2)
      pminf=pfish(2,2)
c
```

```
      write(44,100) xp/geomscale,yp/geomscale,zp/geomscale,
     +time,pfish(2,2)/1000.,sheartot,difu,accel/9.81
      call flush(44)
c
c     write(55,100) xp/geomscale,yp/geomscale,zp/geomscale,
c    +time,umfish,vmfish,wmfish,shearmax
c     call flush(55)
c
      endif
c
 100  format(8e12.4)
c
c write fish node for animations
c
      if(mod(istep,ieveryanim).eq.0) then
c
      if((iensight.eq.1).or.(iensight.eq.3)) then
c Do tecplot output
c
      write(anims,101) 'ANIM',n,'-',istep,'.plt'
      open(22,file=anims)
  101 format(a4,i4.4,a1,i6.6,a4)
c
      write(22,*) 'TITLE="FISHBODY"'
      write(22,*) 'VARIABLES=X,Y,Z,P,DUM,DUM,DUM,DUM'
      write(22,*) 'ZONE T="B", I= ',jxf,',J= ',ixf,' , F=block'
      write(22,*) (((xfish(i,j)/geomscale),j=1,jxf),i=1,ixf)
      write(22,*) (((yfish(i,j)/geomscale),j=1,jxf),i=1,ixf)
      write(22,*) (((zfish(i,j)/geomscale),j=1,jxf),i=1,ixf)
      write(22,*) ((pfish(i,j),j=1,jxf),i=1,ixf)
      write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
      write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
      write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
      write(22,*) ((xfish(i,j),j=1,jxf),i=1,ixf)
      close(22)
      endif
c
      if((iensight.eq.2).or.(iensight.eq.3)) then
c Do Ensight output
c
      write(animse,102) 'displace',n,istep
      open(22,file=animse)
  102 format(a8,i3.3,i6.6)
      write(22,200)
     +((xfish(i,j)/geomscale,yfish(i,j)/geomscale,
     +zfish(i,j)/geomscale,j=1,jxf),i=1,ixf)
  200 format(6(1x,e11.5))
      close(22)
c
      write(animsp,102) 'pressure',n,istep
      open(23,file=animsp)
      write(23,200) ((pfish(i,j),j=1,jxf),i=1,ixf)
c
      endif
c
      endif
c
      return
      end
      subroutine rotation
c
c This soubroutine transforms the velocity field from
c rotational to absolute by adding the runner rpms
c
      include'common_virtual'


      do 100 ilo=1,iblock
      do 100 i=1,ix(ilo)
      do 100 j=1,iy(ilo)
      do 100 k=1,iz(ilo)
c
      u(i,j,k,ilo)=u(i,j,k,ilo)+(y(i,j,k,ilo)*rpm)
      v(i,j,k,ilo)=v(i,j,k,ilo)+(x(i,j,k,ilo)*(-1.)*rpm)

  100 continue

c     open(88,file='ROTATED.DAT')
c     write(88,*) 'TITLE = "ALL"'
c     write(88,*) 'VARIABLES= X, Y, Z,I,U,V,W,P'

c     do 201 ilo=1,iblock
c     write(88,*)
c    +'ZONE T="f", I=',ix(ilo),', J=',iy(ilo),', K=',iz(ilo),' F=block'
c     write(88,*)(((x(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c     write(88,*)(((y(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c     write(88,*)(((z(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c     write(88,*)(((ibl(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),
c    +k=1,iz(ilo))
c     write(88,*)(((u(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c     write(88,*)(((v(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c     write(88,*)(((w(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c     write(88,*)(((p(i,j,k,ilo),i=1,ix(ilo)),j=1,iy(ilo)),k=1,iz(ilo))
c201  continue

      return
      end
c
      subroutine pres_scl_scroll
c
c This soubroutine dimensionalizes the pressure in the
c scroll & vanes and sets it to real full-scale values.
c Dimensionalization procedure provided by:
c Garry Franke and Justin Hall, Voith Hydro Inc.
c
      include'common_virtual'
c

      do 100 ilo=1,iblock
      do 100 i=1,ix(ilo)
      do 100 j=1,iy(ilo)
      do 100 k=1,iz(ilo)
c
      sigmaT=-(p(i,j,k,ilo)-pcfdinlet)/(denw*gi*Href)+
     +(Vinlet**2)/(2.*gi*Href) + hLvsoverH
      pinterm=PrefProto*(sigmaPHW-sigmaT)+Pvapor +
     +denw*gi*(Zref-z(i,j,k,ilo))*geomscale
c
      p(i,j,k,ilo)=pinterm
c
  100 continue
      return
      end

c
      subroutine pres_scl_run
c
c This soubroutine dimensionalizes the pressure in the
c runner and sets it to real full-scale runner values.
c Dimensionalization procedure provided by:
c Garry Franke and Justin Hall, Voith Hydro Inc.
c
      include'common_virtual'
c
      write(*,*) 'Pressure re-dimensionalized'
c

      do 100 ilo=1,iblock
      do 100 i=1,ix(ilo)
      do 100 j=1,iy(ilo)
      do 100 k=1,iz(ilo)
c
      sigmaT=(-p(i,j,k,ilo)/(denw*gi*Href))-HLoverHREF+
     +VSQexover2gHref-VsqDTE
      pinterm=PrefProto*(sigmaP-sigmaT)+Pvapor +
     +denw*gi*(Zref-z(i,j,k,ilo))*geomscale
c
      p(i,j,k,ilo)=pinterm
c
  100 continue
      return
      end
c
      subroutine pres_scl_dt
c
c This soubroutine dimensionalizes the pressure in the
c draft tube and sets it to real full-scale values.
c Dimensionalization procedure provided by:
c Garry Franke and Justin Hall, Voith Hydro Inc.
c
      include'common_virtual'
c

      do 100 ilo=1,iblock
      do 100 i=1,ix(ilo)
      do 100 j=1,iy(ilo)
      do 100 k=1,iz(ilo)
c
      sigmaT=(-p(i,j,k,ilo)/(denw*gi*Href))-HLoverHREF+
     +VSQexover2gHref-VsqDTE
      pinterm=PrefProto*(sigmaP-sigmaT)+Pvapor +
     +denw*gi*(Zref-z(i,j,k,ilo))*geomscale
c
      p(i,j,k,ilo)=pinterm
```

```
c
  100 continue
      return
      end
```