Sliding grid interpolation for Navier Stokes Simulations

Master thesis Report

Jun 2012

Submitted by

Antoine Moise Kinnia Catherine

MSc Computational Mechanics,

Ecole Centrale de Nantes & Swansea University

Under the supervision of

M. Patrick Queutey

Researcher at CNRS

Ecole Centrale de Nantes

For the partial completion of

Master of Computational Mechanics

Ecole Centrale de Nantes

ABSTRACT

This report includes numerical description of Exchange of information's at sliding interfaces by interpolation methods. Communication of data at sliding grid interface is crucial for areas of computing in relative motion. For example, a fixed computational domain and a second rotating inside the first. Such a method is under development for ISIS-CFD solver. Several options of communication at interface are studied. Sub program with Fortan 90 programming language is implemented in the ISIS-CFD solver developed by "DSPM" "Dynamique des Systèmes Propulsifs Marins" of the LHEEA laboratory (Laboratoire de recherche en Hydrodynamique, Energétique et Environement Atmosphérique" de l'Ecole Centrale de Nantes).

ACKNOWLEDGEMENT

To complete this journey several individuals have helped me to whom I would like to express my sincere gratitude.

I am deeply indebted to my supervisor Patrick Queutey, whose support, motivating suggestions and encouragement during all the up and downs that guided me all through my project, without which it would have been hard. I also thank him for giving me this opportunity to do this thesis under his guidance.

Also I would like thank all the members of the group, to give me this opportunity and for their invaluable help and guidance to complete my thesis.

I would like to extend my sincere thanks and love to all my family and all to my friends, who supported me morally and accompanied me always. Last but not the least, I thank god for all his guidance and grace.

Contents

CHAPTER I - INTRODUCTION

1.1. ISIS CFD RANSE SOLVER

ISIS-CFD is an incompressible flow solver, that uses incompressible unsteady Reynolds-averaged Navier Stokes equation (RANSE). The ISIS-CFD flow solver was developed by "DSPM" "Dynamique des Systèmes Propulsifs Marins" of the LHEEA laboratory (Laboratoire de recherche en Hydrodynamique, Energétique et Environement Atmosphérique" de l'Ecole Centrale de Nantes). It's commercially distributed by Numeca in a software package named Fine/Marine including Hexpress, a hexahedral unstructured mesh generator, ISIS-CFD flow solver, and a postprocessor CFView.

The solver is based on the finite volume method to build the spatial discretization of the transport equations. The face-based method is generalized to two-dimensional, rotationally symmetric, or three-dimensional unstructured meshes for which non-overlapping control volumes are bounded by an arbitrary number of constitutive faces. The velocity field is obtained from the momentum conservation equations and the pressure field is extracted from the mass conservation constraint, or continuity equation, transformed into a pressure equation. Free-surface flow is simulated with a multi-phase flow approach: the water surface is captured with a conservation equation for the volume fraction of water, discretized with specific compressive discretization schemes (Queutey P, 2007)

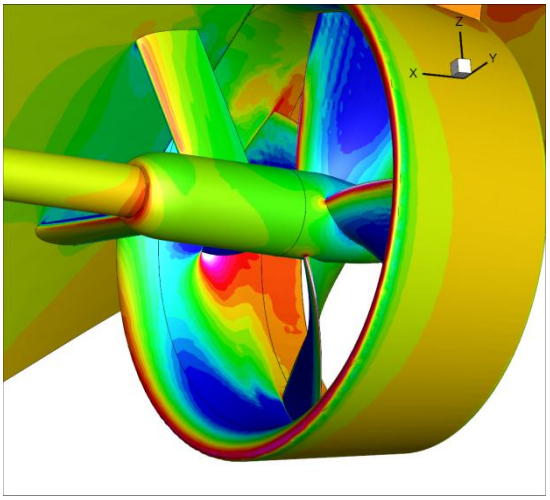
The accuracy of the ISIS-CFD RANS solver is obtained using advanced features in turbulence modeling (i.e. nonlinear model and rotation correction, high and low Reynolds models, etc.), free-surface capturing methods with specific compressive discretization schemes for the concentration transport equation. This leads to a sharp capturing of the density discontinuity between air and water (NUMECA International)

The solver also incorporates a mesh deformation algorithm for unstructured grids. Furthermore, the solver can handle the free movement of the solid bodies with 6 degrees of freedom. The unique adaptive grid refinement method (refinement - de-refinement, parallel, unsteady and with load balancing) opens a wide range of opportunities: capturing the free surface in time without the need of a fine mesh in the complete domain. Also, other refinement sensors such as pressure gradient and vorticity exist to locally refine the mesh.

1.2. SCOPE AND MOTIVATION

When solving numerical problems involving large structures, for example turbo machineries, usually the mesh generation is performed separately and combined together to perform the 3D simulations. This may involve the interactions between moving and sliding grid. The interactions of moving entities with fluid flow are common to numerous engineering and biomedical applications. A major concern of such applications is to fit all meshes with conformal matching interfaces without compromising the geometry.

Especially in the turbo machineries, there are many complex problems associated like accurate prediction of forces and wake flow, physical Modélisation such as cavitation, turbulence and ventilation, propeller hull interaction. Among all the complexity mentioned earlier, one such study currently under research is to ensure the good communication at the interface of the sliding surface of ship propeller (Figure 1). Here the propeller is modeled with the moving grid which is rotating inside a fixed grid (Figure 2).



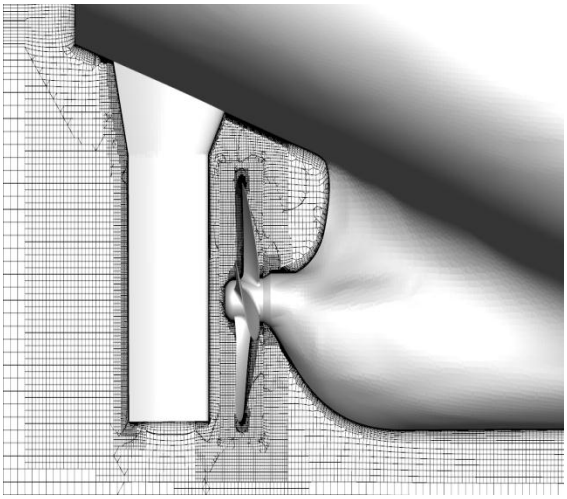


Figure 1: Distribution of the surface pressure **Figure 2: Grid view across the propeller with sliding interface about a ducted propeller**

The objective of the work is to propose an interpolation scheme that helps to recalculate the mass fluxes at every time step between the moving and fixed grid, so that the coupling between the fixed and moving grids are enhanced.

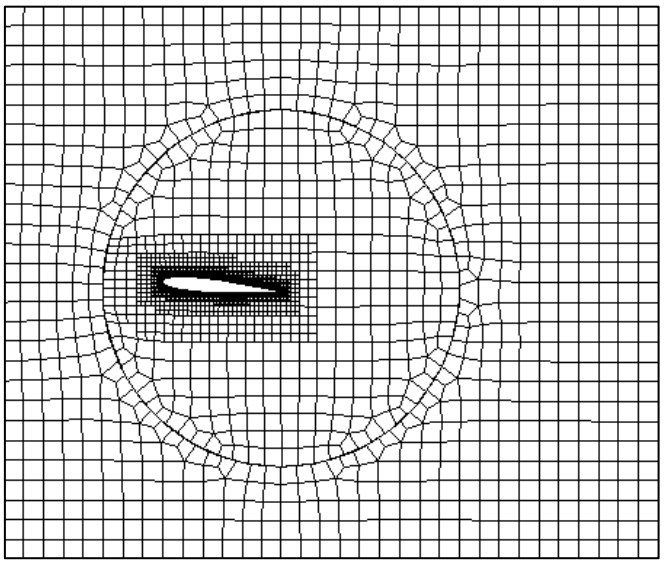


Figure 3: Non conformal mesh interface in 2D test case

In (Martin Beaudoin, Hrvoje Jasak, 2008), the authors have proposed a method of adding an weighting factor that is termed as Generalized grid interface (GGI) weighting factor, that removes the requirement to adjust the topology of the mesh at the two non-conformal mesh interfaces. The authors describe a method of adding weighting factors both to master and slave surface. These weighting factors are basically the percentage of surface intersection between two overlapping faces. These values obviously need to be greater than zero; otherwise this would mean no intersection between two given faces and clearly indicates a failure of the neighborhood determination algorithm. This calls for an efficient face to face intersection algorithm. To overcome these problems, the authors have used Sutherland-Hodgman algorithm which is designed to handle only convex polygons, which makes this algorithm only specific to OpenFOAM meshes. This calls a need to investigate on a method that is based on global approach.

1.3. STRUCTURE OF THE REPORT

The motive of the current work is to investigate on different scattered data interpolation methods available on literatures and how to efficiently we can implement one such method for our applications. The conservative nature of the interpolation which is an another point of interest for the sliding interface in the context of the finite volume method is not focused in this project.

The objective of this thesis work can be summarized in three parts

1. Acquire knowledge of interpolation methods through literature review.
2. Investigate on the possibilities of incorporating selected interpolation method by implementing in the Solver by a subroutine

3. Determine which interpolation method is best to ensure a good communication of data's between the moving grid and fixed grid interface

In the next chapter a detailed summary of investigated interpolation methods are presented. Literature review based on three domains namely Medical imaging, Geostatistical interpolation in Geophysical sciences and Meshless methods based on Interpolation methods in Computational mechanics field, in which scattered data interpolation is of major help is discussed.

From the detailed literature review, four methods namely, Weighted interpolation methods like Inverse distance weighted interpolation, Modified Shepard Interpolation, Artificial Network Method with Linear, Multiquadratic and Quadratic functions, are implemented as a subroutine in the solver using Fortran Compiler. This implementation is discussed in Chapter III.

Among the interpolation methods implemented, Inverse distance weighted interpolation method being the simplest one, is analyzed explicitly based on power functions, different time step size and reduced pressure relaxation factors. Artificial neural network interpolation being the complicated one, the difficulties faced in implementing the code and the sensitivity of the delta function are discussed in Chapter IV.

As the scope of this research is extensive, further investigations on prospective methods like implementing Artificial neural network with different functions like exponential function, least square interpolation function and differential equation interpolant can be performed and results can be validated with real applications. This future prospective of the thesis are discussed in the final chapter.

CHAPTER II - SCATTERED DATA INTERPOLATION

The goal of scattered data interpolation is to reconstruct the underlying function that may be evaluated at any desired set of points. By this the smooth propagation of information associated with the scattered data onto all positions in the domain can be done.

There are three types of scattered data (Nielson, R. Franke and G.M., 1980) physical quantities, experimental results and computational values. In most of scientific and engineering applications we come across scattered data and the ways to deal with it according to the necessity of the application. For example, the most said example is the Geostatistical field where the nonuniformly distributed data's are interpolated to predict the weather conditions by using Spatial interpolation. Spatial interpolation is the process of using points with known values to estimate values at other unknown points. For example, to make a precipitation (rainfall) map of a region, we need to find enough evenly spread weather stations to cover the entire region. Spatial interpolation can estimate the temperatures at locations without recorded data by using known temperature readings at nearby weather stations. This type of interpolated surface is often called a statistical surface. Elevation data, precipitation, snow accumulation, water table and population density are other types of data that can be computed using interpolation. Finite element solutions of partial differential equations is well known example for computational engineers, along with data approximations in computer graphics and computer vision. One another prominently growing domain is medical field. For instance, in the medical imaging scattered data interpolation is very essential to construct a closed surface for CT or MRI images.

IMAGE interpolation has many applications in computer vision. It is the first of the two basic resampling steps and transforms a discrete matrix into a continuous image. Subsequent sampling of this intermediate result produces the resampled discrete image. Resampling is required for discrete image manipulations, such as geometric alignment and registration, to improve image quality on display devices or in the field of lossy image compression wherein some pixels or some frames are discarded during the encoding process and must be regenerated from the remaining information for decoding. Therefore, image interpolation methods have occupied a peculiar position in medical image processing. They are required for image generation as well as in image post-processing. In computed tomography (CT) or magnetic resonance imaging (MRI), image reconstruction requires interpolation to approximate the discrete functions to be back projected for inverse Radon transform. In modern X-ray imaging systems such as digital subtraction angiography (DSA), interpolation is used to enable the computer-assisted alignment of the current radiograph and the mask image. Moreover, zooming or rotating medical images after their acquisition often is used in diagnosis and treatment, and interpolation methods are incorporated into systems for

computer aided diagnosis (CAD), computer assisted surgery (CAS), and picture achieving and communication systems (PACS).

Despite of its wide application, the scattered data interpolation still remains a difficult and computationally expensive problem. There are many authors who have proposed an efficient algorithms (WILLIAM I. THACKER, 2009), but it's always a compromise between the accuracy and the computational cost. With the accurate models, the problem becomes more complex and hence impacting the computational cost. Or the smoothness of the solution is affected in simple models.

2.1. Previous Work

2.1.1. Interpolation Methods

In the early years, a great variety of methods with confusing naming can be found in the literature. In 1983, Parker, Kenyon, and Troxel published the first paper entitled "Comparison of Interpolation Methods" (J.A.Parker, 1983). Parker et al. pointed out that, at the expense of some increase in computing time, the quality of resampled images can be improved using cubic interpolation when compared to nearest neighbor, linear, or B-spline interpolation. Maeland named the correct (natural) spline interpolation as B-spline interpolation and found this technique to be superior to cubic interpolation (Maeland, 1988)

2.1.2. Interpolation in Medical Image Processing

Image interpolation methods are as old as computer graphics and image processing. In the early years, simple algorithms, such as nearest neighbor or linear interpolation, were used for resampling. In more recent reports, not only hardware implementations for linear interpolation and fast algorithms for B-spline interpolation or special geometric transforms , but also nonlinear and adaptive algorithms for image zooming with perceptual edge enhancement have been published. However, smoothing effects are most bothersome if large magnifications are required. In addition, shape-based and object-based methods have been established in medicine for slice interpolation of three-dimensional (3-D) data sets. In 1996, Appledorn presented a new approach to the interpolation of sampled data. His interpolation functions are generated from a linear sum of a Gaussian function and their even derivatives. Kernel sizes of 8 8 were suggested. Again, Fourier analysis was used to optimize the parameters of the interpolation kernels. Contrary to large kernel sizes and complex interpolation families causing a high amount of computation, the use of quadratic polynomials on small regions was recommended by Dodgson in 1997.

A detailed literature survey of progress in interpolation methods for image processing was given by (Thomas M. Lehmann, 1999). The authors have also compared 1) truncated and windowed sinc; 2) nearest neighbor; 3) linear; 4) quadratic; 5) cubic B-spline; 6) cubic; g)

Lagrange; and 7) Gaussian interpolation and approximation techniques with kernel sizes from 1 X 1 up to 8 X 8. The comparison was done by: 1) spatial and Fourier analyses; 2) computational complexity as well as runtime evaluations; and 3) qualitative and quantitative interpolation error determinations for particular interpolation tasks which were taken from common situations in medical image processing. They have concluded that all interpolation methods smooth the image more or less. Images with sharp-edged details and high local contrast are more affected by interpolation than others. Artificial Neural network is gaining popularity due to its accuracy in predicting the missing variables. For the first time, Arjun Akkala et al (Arjun Akkala, 2011,) used comparison of the ANN model with conventional interpolation techniques to find radon concentrations a region, and showed that the proposed technique results in relatively better accuracies

2.1.3. INTERPOLATION IN GEOPHYSICAL SCIENCES

Spatial interpolation, or the estimation of a variable at an unsampled location using data from surrounding sampled locations, has great importance in all geophysical sciences. Geophysical disciplines all share the impossibility of sampling accurately enough, in space or in time, their object of study.

Names given to spatial interpolation can differ depending on the field of study. In meteorology and oceanography, the process of defining the best estimate of the state of the system at a given time (using both observations and the a-priori knowledge of physical processes) goes under the name of spatial analysis, or objective analysis to distinguish it from the earlier manual charting methods.

As per (Hou, Andrews 1978), piecewise polynomial especially B-splines are often very appropriate to define a surface in geophysical applications. Once the parameters of the model known by adjustment over the observed values, it provides a value for any geographical point P for the area under concern. Least-square fitting of surfaces of polynomial type on a global or local scale, thin plate method or Hsieh-Clough-Tocher method, are examples of such methods (Hutchinson et al. 1984; Hulme et al. 1995). One of their advantages is the degree of continuity of the derivatives of the estimated field.

Kriging is probably the most widely used technique in geostatistics to interpolate data. It was formalized in the sixties by the French engineer George Matheron (Matheron, 1963) after the empirical work of Danie G. Krige (Krige, 1951). In the early 1960s, Barnes introduced a new convergent weighted-averaging interpolation scheme based on the hypothesis that the spatial distribution of the atmospheric variables can be described by a Fourier integral representation (Barnes, 1964). Now, Barnes's technique is usually discussed in the context of the Successive Corrections Method (Daley, 1991).

2.1.4. INTERPOLATION IN COMPUTATIONAL MECHANICS:

Implementation of General grid interface was proposed by (Martin Beaudoin, 2008) for OpenFoam. Generalized Grid Interface (GGI) uses a set of weighting factors to properly balance the flux at grid interfaces and thus removing the requirement to adjust the topology of the mesh at the interface. The authors used the generalized grid interface for turbo machinery simulations which is validated on numerous test cases, and provides satisfactory results.

Meshless method to solve the partial differential equations are equally gaining importance, and lots of papers are being published in this domain. This application in partial differential equations was encouraged by the pioneer work done by Kansa (Kansa, 1990) Kansa's algorithm was similar to finite difference method (FDM), but node distribution was completely unstructured. This was overcome by collocation methods proposed later. One another method called, h-p cloud method, proposed by (C Armando Duarte, 1996) is also based on radial basis functions. This method can be applied to arbitrary domains and only scattered data can be used to employ this method. It was proved by the authors that this method exhibits a very high rate of convergence than traditional h-p finite element methods. Further do this, lots of papers been published using approximating interpolation functions. Another pioneering work was by (J. G. Wang, 2002), in which the authors have employed a meshless methos based on combining radial and polynomial basis functions. This helped in implementing essential boundary conditions much easier than the Meshless methods based on the moving least squares approximation. This was possible since the singularity associated with the polynomial functions were overcome by the radial functions and this non singularity aids in constructing a well performed shape functions.

In recent years artificial neural networks (ANN) have been applied as well for such purposes. Artificial Neural Networks have been designed to model the processes in the human brain numerically. These mathematical models are used today mainly for classification problems as pattern recognition. In the recent years a large number of different neural network types have been developed, e.g. the multi-layer perceptron, the radial basis function network, networks with self-organizing maps and recurrent networks. A detailed review of this method is given by Thomas Most in his report (Most). According to his report, Neural networks (ANN) have been applied in several studies for stochastic analyses, e.g. in Papadrakakis et al. (1996), Hurtado and Alvarez (2001), Cabral and Katafygiotis (2001), Nie and Ellingwood (2004a), Gomes and Awruch (2004), Deng et al. (2005) and Schueremans and Van Gemert (2005). In these studies the structural uncertainties in material, geometry and loading have been modeled by a set of random variables. A reliability analysis has been performed either approximating the structural response quantities with neural networks and generating ANN based samples or by reproducing the limit state function by an ANN approximation and decide for the sampling sets upon failure without additional limit state

function evaluations. The main advantage of ANN approximation compared to RSM is the applicability to higher dimensional problems, since RSM is limited to problems of lower dimension due to the more than linearly increasing number of coefficients. In Hurtado (2002) firstly a neural network approximation of the performance function of uncertain systems under the presence of random fields was presented, but this approach was applied only for simple one-dimensional systems. A further application in engineering science is the identification of parameters of numerical models e.g. of constitutive laws. In Lehky and Novhak (2004) the material parameters of a smeared crack model for concrete cracking were identified.

2.2. PROBLEM DEFINITION

The scattered data interpolation problem can be defined as, given a *n* set of irregularly distributed points,

$$P_i = (x_i, y_i), \quad i = 1,2,3, \dots \dots, n \qquad (1)$$

And scalar values F_i associated with each point satisfying $F_i = F(x_i, y_i)$ for some underlying function $F(x, y)$, look for an interpolating function $\bar{F} \approx F(x, y)$, such that for $i = 1,2,3, \dots, n)$

$$\bar{F}(x_i, y_i) = F_i \qquad (2)$$

We assume that all the points (also referred to as nodes or mesh points) are distinct, and that all the points are not collinear. This formulation can be generalized to higher dimensions but in this article, we will concentrate on the two-dimensional case.

2.3. TYPES OF INTERPOLATION METHODS

There are different classes of interpolation methods such as geometrical nearness (e.g. the Voronoi approach), statistical methods (e.g. natural neighbor interpolation, weighting inverse distances, Kriging), using basis functions (e.g. trend surface analysis, regularized smoothing spline with tension, method of local polynomials) and the Artificial Neural Networks (ANN) method.

Though there numerous literature available on large family of interpolation methods, we specific our study to the point data interpolation for our requirement and the methods that concentrate on a small sample data.

2.3.1. INTERPOLATION BY WEIGHTED AVERAGES

Usage of weighted averages to interpolates dates back to early stage of Computer era. Weighting is assigned to sample points through the use of a weighting coefficient that controls how the weighting influence will drop off as the distance from new point increases.

The greater the weighting coefficient, the less the effect points will have if they are far from the unknown point during the interpolation process. As the coefficient increases, the value of the unknown point approaches the value of the nearest observational point.

Mathematical Definition:

It defines an interpolating function that is the weighted average of the value at the mesh points.

$$F(x,y) = \sum_{i=1}^{n} w_i f_i \qquad (3)$$

Where n is the number of scatter points in the set, f_i are the prescribed function at the scatter points (e.g. the data set values), and w_i are the weight functions assigned to each scatter point.

2.3.1.1. INVERSE DISTANCE WEIGHTED INTERPOLATION

One of the most commonly used techniques for interpolation of scatter points by weighted average is inverse distance weighted (IDW) interpolation. Inverse distance weighted methods are based on the assumption that the interpolating surface should be influenced most by the nearby points and less by the more distant points.

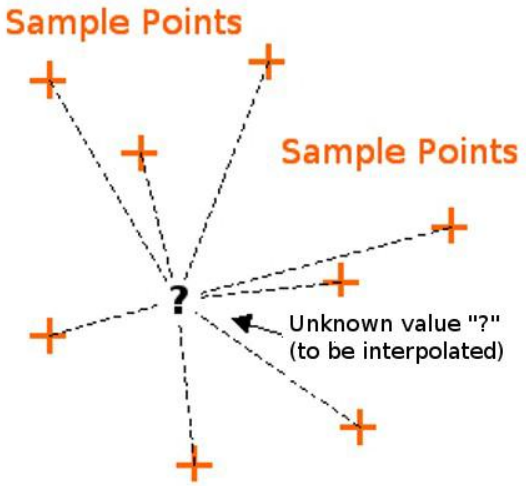


Figure 4: Inverse Distance Weighted interpolation based on weighted sample point distance

The interpolating surface is a weighted average of the scatter points and the weight assigned to each scatter point diminishes as the distance from the interpolation point to the scatter point increases. Several options are available for inverse distance weighted interpolation. The Shepard's Method is one of the earliest techniques used to generate interpolants for scattered data. By Shepard method, the weight function takes the form,

$$w_i = \frac{d_i^{-p}}{\sum_{j=1}^{n} d_j^{-p}} \qquad (4)$$

where p is an arbitrary positive real number called the weighting exponent . The weighting exponent can be modified and can be compared which was done in the implementation stage. d_i is the distance from the scatter point to the interpolation point.

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \qquad (5)$$

Where (x, y) are the co-ordinates of the interpolation point and (x_i, y_i) are the coordinates of each scatter point. The weight function varies from a value of unity at the scatter point to a value approaching zero as the distance from the scatter point increases. The weight functions are normalized so that the weights sum to unity.

This method is global because the evaluation of the interpolant requires the evaluation of a function on all given mesh points. As described by the authors (Jin Li, 2008) in their review, the main factor affecting the accuracy of IDW is the value of the power parameter (Isaaks and Srivastava, 1989). The choice of power parameter and neighbourhood size is arbitrary (Webster and Oliver, 2001). The most popular choice of p is 2 and the resulting method is often called inverse square distance or inverse distance squared (IDS). The power parameter can also be chosen on the basis of error measurement (e.g., minimum mean absolute error, resulting the optimal IDW) (Collins and Bolstad, 1996). The smoothness of the estimated surface increases as the power parameter increases, and it was found that the estimated results become less satisfactory when p is 1 and 2 compared with p is 4 (Ripley, 1981). IDW is referred to as “moving average” when p is zero (Brus et al., 1996; Hosseini et al., 1993; Laslett et al., 1987), “linear interpolation” when p is 1 and “weighted moving average” when p is not equal to 1 (Burrough and McDonnell, 1998). However, this form of the weight functions accords too much influence to data points that are far away from the point of approximation and may be unacceptable in some cases.

2.3.1.2. MODIFIED SHEPARD’S INTERPOLATION METHOD

Franke and Nielson (Franke. R., 1980) developed a modification that eliminates the deficiencies of the original Shepard’s method. They modified the weight function $w_k(x)$to have local support and hence to localize the overall approximation, and replaced $f(x)$ with a suitable local approximation $P_k(x)$. This method is called the local modified Shepard method and has the general form

$$F(x, y) = \frac{\sum_{k=1}^{n} w_k(x) P_k(x)}{\sum_{k=1}^{n} w_k(x)} \qquad (6)$$

where $\mathrm{P_k(x)}$ is a local approximant to the function $\mathrm{f(x)}$ centered at $\mathrm{x(k)}$, with the property that $\mathrm{P_k(x_k) = f_k}$. The choice for the weight functions $\mathrm{w_k(x)}$ used by Renka (J., 1988) was suggested by Franke and Nielson (Franke. R., 1980) and is of the form

$$W_k(x) = \left[\frac{\left(R_w^k - d_k(x)\right)_+}{R_w^k d_k(x)}\right]^2 \quad (7)$$

where $d_k(x) = \|x - x_k\|_2$ is the Euclidean distance between x and x_k and the constant $R_w^k > 0$ is a radius of influence about the point x_k chosen just large enough to include N_w points. The data around x_k only influences $f(x)$ values within this radius.

There are several variations of the original Shepard algorithm based on polynomial and trigonometric functions for $P_k(x)$.

The polynomial function P_k is written as a Taylor series about the point x_k with constant term $f(k) = P_k(x^k)$. and coefficients chosen to minimize the weighted sum of squares error

$$\sum_{\substack{i=1 \\ i \neq k}}^{n} w_{ik}\left[P_k(x^i) - f_i\right]^2 \quad (8)$$

with weights

$$W_{ik} = \left[\frac{\left(R_p^k - d_i(x^k)\right)_+}{R_p^k d_i(x^k)}\right]^2 \quad (9)$$

And $R_p^k > 0$ defining a radius about x^k within which data is used for the least squares fit. R_w and R_p are taken by Franke and Nielson (Franke. R., 1980) as

$$R_w = \frac{D}{2}\sqrt{\left(\frac{N_w}{n}\right)}, R_p = \frac{D}{2}\sqrt{\left(\frac{N_p}{n}\right)} \quad (10)$$

Where $D = \max_{i,j}\|x(i) - x(j)\|_2$ is the maximum distance between any two data points, and N_w and N_p are arbitrary positive integers. The constant values for R_w and R_p are appropriate assuming uniform data density.

Though the method above proposed by Thacker et al. (WILLIAM I. THACKER, 2009) provides statistically robust least squares fits as required by most applications like local approximations in image processing, linear interpolation proposed by (Franke. R., 1980) is implemented for our application. This simple model is initially implemented just to compare the solution with other methods, as theoretically this model cannot suit our application since

its used to interpolate the function based on set of data at radial domain. According to (Franke. R., 1980), the weight function for Modified Shepard interpolation takes the form

$$w_i = \frac{\left[\frac{R-d_i}{Rd_i}\right]^2}{\sum_{j=1}^{n}\left[\frac{R-d_j}{Rd_j}\right]^2} \qquad (11)$$

Where d_i is the distance from the interpolation point to scatter point i, R is the distance from the interpolation point to the most distant scatter point, and n is the total number of scatter points. This equation has been found to give superior results to the classical Shepard equation (Franke. R., 1980)

The weight function is a function of Euclidean distance and is radially symmetric about each scatter point. As a result, the interpolating surface is somewhat symmetric about each point and tends toward the mean value of the scatter points between the scatter points.

Both the Shepard's method and Modified Shepard Method has been used extensively because of its simplicity.

2.3.2. ARTIFICIAL NEURAL NETWORK INTERPOLATION

Artificial Neural Networks (ANNs) are information processing systems that have the ability to implement new information formation and discovery automatically using the mode of learning of human brain and neural biology. ANNs are generally used for classification, prediction, identification, recognition and interpolation problems. The basic processing elements of an ANN are the neurons (units). The multi-layer ANN model is typically composed of three parts: input, one or many hidden layers or basis functions, and an output layer Figure 5.

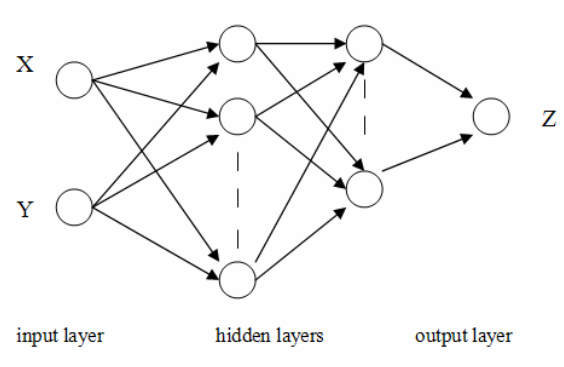


Figure 5: Artificial neural network

In the approach discussed in this report, the outputs of the network are determined by linear combinations of these basis functions. The effectiveness of basis functions from Radial basis functions and the Artificial neural network methodology for non-radial applications are

combined together and the results are analyzed. Accordingly, the interpolant will be of the form,

$$f(x) = \sum_{j=1}^{M} w_j \varphi_j(x) \quad (12)$$

Where where $f(x)$ denotes the required output value and w_j represents the synaptic weight from the jth basis function (or hidden node) to the output and $\varphi_j(x) = \|x - x_j\|$
Basis function neural network training has the following two-stage procedure. In the first stage to find the weight function, the interpolant is forced to undergo a condition,

$$f(x_i) = \sum_{i=1}^{M} w_j \varphi_j(x_i) \quad (13)$$

Where, $\varphi_j(x_\mathrm{i}) = \|x_\mathrm{i} - x_j\|$. As the functions $f(x_i)$ from the given set of data is known, weight function w_j can be found from this conditions is back substituted in equation 12 to find the required interpolation function.

Several forms of a basis function have been considered, the most common being the Gaussian, and recently, Multiquadratic and Quadratic basis functions are also gaining popularity.

Gaussian function,

$$\varphi_j(x_\mathrm{i}) = e^{-d_{ij}} \quad (14)$$

Multiquadratic function,

$$\varphi_j(x_\mathrm{i}) = \sqrt{\delta^2 + d_{ij}} \quad (15)$$

Quadratic function,

$$\varphi_j(x_\mathrm{i}) = \delta^2 + d_{ij} \quad (16)$$

Inverse Quadratic function,

$$\varphi_j(x_\mathrm{i}) = \frac{1}{\delta^2 + d_{ij}} \quad (17)$$

Inverse Multi Quadratic function,

$$\varphi_j(x_\mathrm{i}) = \frac{1}{\sqrt{\delta^2 + d_{ij}}} \quad (18)$$

Where,

$$d_{ij} = \left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2 \qquad (19)$$

For our case Quadratic and multiquadratic cases are implemented and results are compared. Though there is lot of papers available on this method, only very few articles are found explaining the δ properties. Its used to smooth the function, which is an mere mathematical approximation.

2.3.3. KRIGING INTERPOLATION

The Kriging method assumes that the distance or direction between sample points reflects a spatial correlation that can be used to explain variation in the surface. Kriging is a multi-step process; it includes exploratory statistical analysis of the data, variogram modeling, creating the surface, and optionally, exploring a variance surface. Inverse Distance Weighted and Spline are referred to as deterministic interpolation methods because they are directly based on the surrounding measured values or on specified mathematical formulas that determine the smoothness of the resulting surface. A second family of interpolation methods consists of geostatistical methods such as Kriging, which are based on statistical models that include autocorrelation (the statistical relationship among the measured points). Because of this, not only do these techniques have the capability of producing a prediction surface, but they can also provide some measure of the certainty or accuracy of the predictions. There are two important Kriging methods used; Ordinary Kriging and Universal Kriging. Ordinary Kriging is the most general and widely used of the Kriging methods. It assumes the constant mean is unknown. Universal Kriging assumes that there is an overriding trend in the data and it can be modeled by a deterministic function, a polynomial. This polynomial is subtracted from the original measures points, and the autocorrelation is modeled from the random errors. Once the model is fit to the random errors, before making a prediction, the polynomial is added back to the predictions to give a meaningful result.

As it gives more accurate prediction for 3D case, rather than 2D case, this method was not tested for our application, as it requires robust algorithm for computations.

CHAPTER III IMPLEMENTATION

In this chapter the method of interface interpolation implementation in the ISIS-CFD solver is explained. Investigation of related exchange of information between the sliding Grids by a appropriate methodology is analyzed and the development of communication methodologies based on sliding grid to enable the connection from a fixed reference frame to the relative flow in the rotating system of reference based on different interpolation functions and how the different methods of interpolation executed in the code was discussed.

To get into the subject in detail, one can find it very useful to know the basic grid structure in computational fluid dynamics and ways of interactions between neighbor cells.

Grid: It is the real entity of the mesh and provides residence for variables. Due to the pressure decoupling phenomenon, it is necessary to maintain several groups of grids for storing different variables. The mesh composited of these grids is called staggered mesh.

- Cell-centered grid
- Edge-centered grid
- Vertex grid

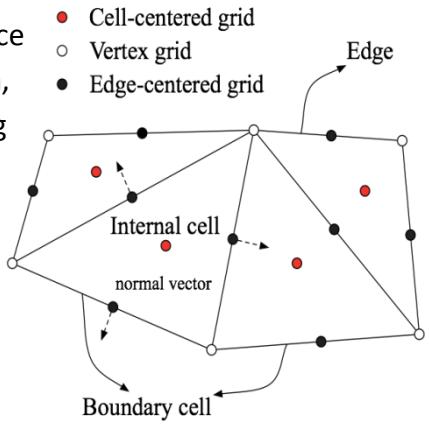


Figure 6: Basic elements of mesh

Cell: It describes the connectivity's among grids by grouping the indices of vertex grids. Cell is surrounded by edges and has several neighbor cells.

- Internal cell
- Boundary cell: It is equipped with unit normal vector to judge whether a point is inside or outside the domain. The dimension of it is less than the internal cell one. And it has the boundary class attribute which is used for boundary condition.

Edge: It may be a point (1D), line (2D), or face (3D). Vertex indices are stored, and also the unit normal vector.

3.1. RECONSTRUCTION ON FACES

As along with the other complications involved in the Computational fluid dynamics calculations of turbo machinery applications as discussed in the previous chapters, its vital to get an accurate description of a density discontinuity with a coherent treatment for the variable coefficient pressure equation in presence of interfaces where both the variable coefficients and the solution itself may be discontinuous. To overcome this problem a specific numerical algorithm and discretization schemes were proposed by (Queutey P, 2007) based on reconstruction of fluxes at faces (Figure 8). The same methodology can be used for grid interface at moving and fixed grid domain, where there is discontinuity in the solution.

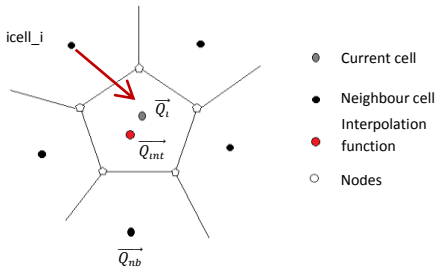


Figure 8: General outline of Grids and interpolation function

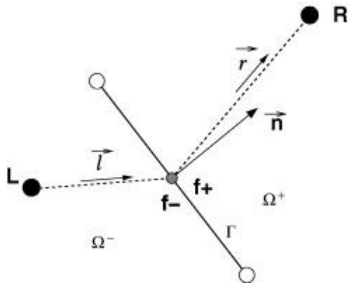


Figure 8: Face reconstruction

In the face reconstruction, according to the authors, (Queutey P, 2007) the face fluxes are computed from the quantities in the cell centres reconstructed to the faces. For the diffusive fluxes and the coefficients in the pressure equation, the quantities on a face and the normal derivatives are computed with central schemes using the L and R cell centre states; if these centres are not aligned with the face normal, then non-orthogonal corrections are added which use the gradients computed in the cell centres. For the convective fluxes, the AVLSMART scheme Przulj and Basara (2001) was used in the NVD context for unstructured, where limited schemes are constructed based on a weighted blending of the central difference scheme and an extrapolation using the gradient in the upwind cell. For our case, rather to use the fluxes at the inside cell center, we create a ghost point which is linear to the outside cell and whose functions are interpolated (Figure 8)

3.2. TEST CASE

2D test case with farfield velocity of 10m/s that uses the K-w SST model, is used to study the identified interpolation methods used for analyze the mass fluxes transfer at the sliding grid interface (Figure 9). ½ Sinusoidal ramp profile is used for the motion parameter definition of the hydrofoil. Water density is 998.4 kg/m3. Dynamic viscosity of the fluid is set at 0.00104362 kg/ms and with the reference velocity of 1 m/s that gives the Reynolds no of 9.5667^5. Calculations are done at timestep size of 0.01s of total time 10s and for specific cases, different timestep sizes of 0.002s, 0.005s, 0.001s are used .

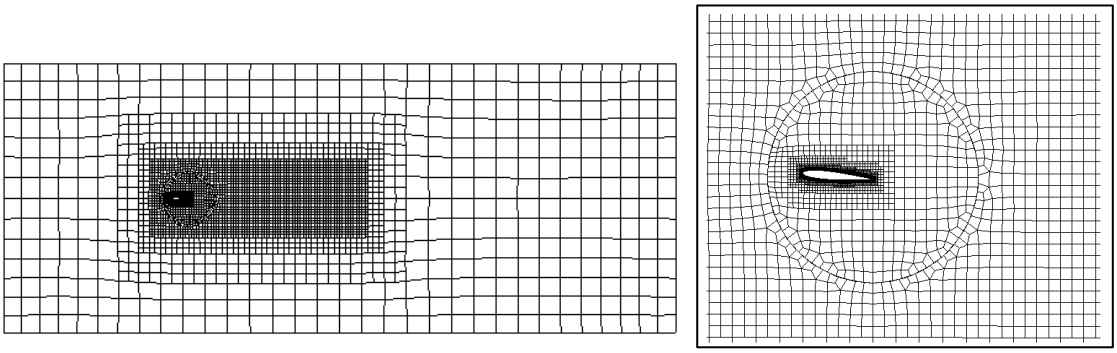


Figure 9: Meshed 2D test case

3.3. PROBLEM DEFINITION

Let icell_i be the current cell, whose flow function Q_i has to be interpolated Q_{int} based on the neighbor cells and the external cell on the other side of the interface icell_e (Figure 10)

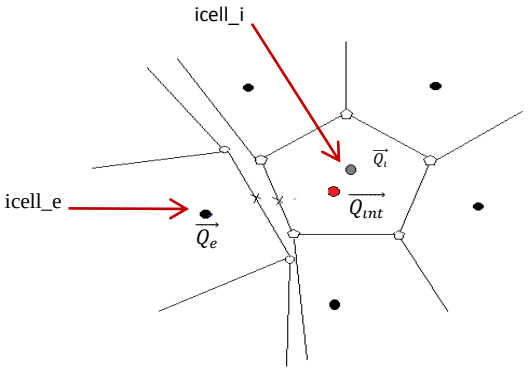


Figure 10: Interpolation function at sliding grid interface

Complexity with moving cell:

With the automatic grid refinement and with changing time space, the cell reference will be updated for every time step. (Figure 11) Hence to implement the interpolation scheme and to find the function in the moving grid interface, it's necessary to address the neighbor cells associated with it in an array.

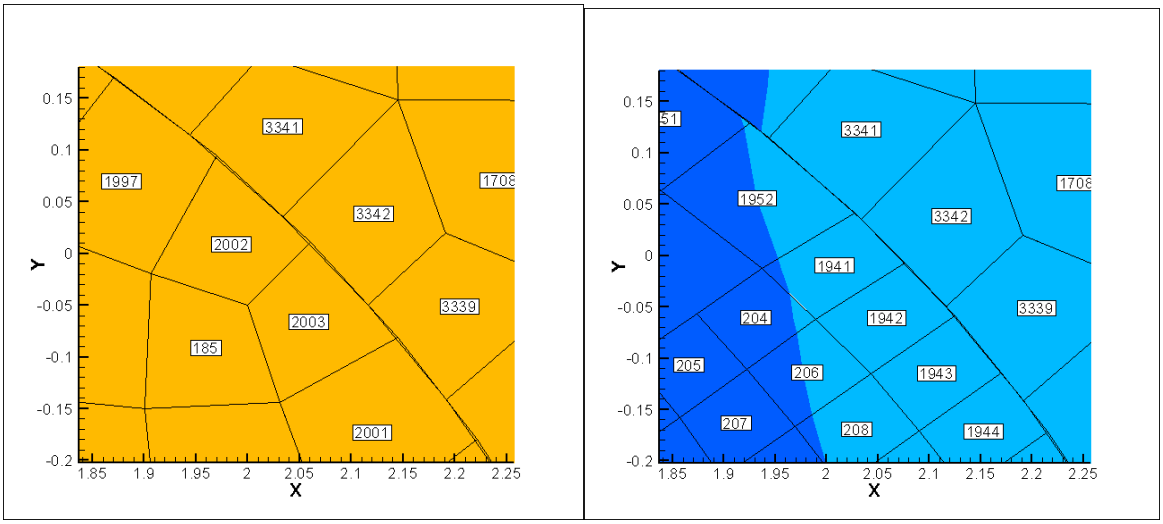


Figure 11: Different neighbors with change in time

The neighbors of the current cell (already implemented in the Solver, and called into the subroutine to perform the calculation), is calculated using existing algorithm developed by the ISIS CFD team. The main interest of this work is to propose a interpolation method, which can give an accurate mass transfer at interfaces. Hence only the array definition is presented in this report for implementation purpose.

This was done with the help of Index connector array (Indcon_CC) and Index pointer array (Ipnt_CF) arrays. These arrays gives the correlation between the connection between cells and faces or between the cells and themselves.

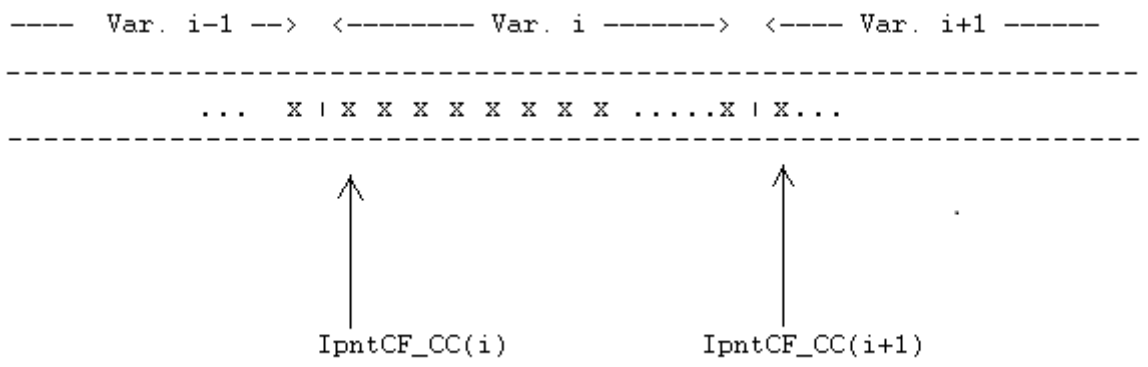


Figure 12: Indcon_CC array and its pointers

If i^{th} cell index denotes the current cell, then the pointer of its neighbor cells are stored in the subsequent cells, which is clearly depicted in the Figure 12

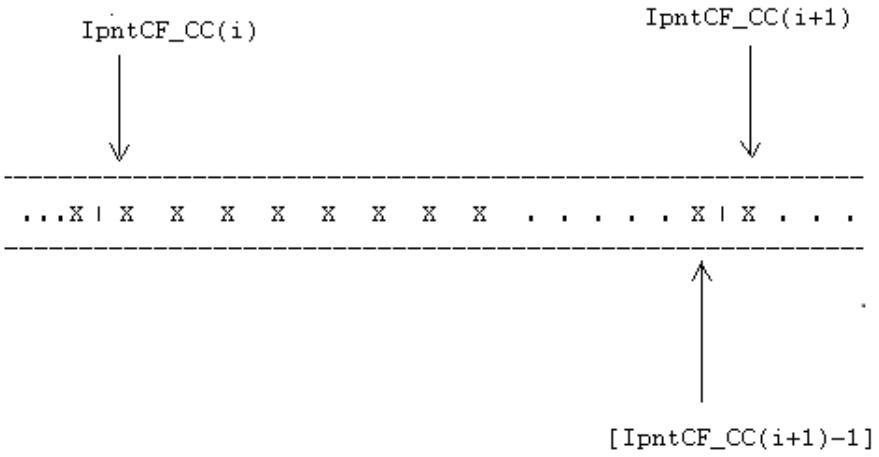


Figure 13: IpntCF_CC array and its pointers

If IpntCF_CC(i) gives the index of the current cell and IpntCF_CC(i+1) gives the index of the next cell number then [IpntCF_CC(i+1)-1] gives the index of the last neighbor of the current cell or the last data of the current cell Figure 13.

3.4. IMPLEMENTATION

Fortran 90 programming language is used to create the subroutines for ISIS-CFD Solver. Dynamic libraries are set locally to call for the main program and computations are run parallel in 2 CPU's. Global naming conventions are used for input and output file arguments. Input arguments called in for the sub routine are array's IpntCF_CC , Indcon_CC, current cell coordinates, (X_c, Y_c, Z_c) and mass function Q. Along with the scalar quantities, ghost point coordinates $(x_{gint}, y_{gint}, z_{gint})$, current cell index $icell\_i$, outer cell index $icell\_e$, and the output variable Q_{int}. The algorithm for the different interpolation method is discussed below.

3.4.1. INVERSE DISTANCE WEIGHTED INTERPOLATION

Initialization

1) Determine weighting function at icell_i

$$d_i = 1.0/\left(\left\|\left(X_c(\text{icell\_i}) - x_{gint}\right)\right\|_2\right)^{pow} \quad (20)$$

Where for power, value of 1,2 and 4 were compared

2) Determine interpolant function at icell_i

$$Q_{int} = d_i * Q(\text{icell\_i}) \quad (21)$$

Calculation:

Start the loop

3) calculate the weighting function with the Euclidean distance from each neighbors to the ghost cell

$$d_i = 1.0/\left(\left\|\left(X_c(\text{ivnb}) - x_{gint}\right)\right\|_2\right)^{pow} \quad (22)$$

where ivnb =IndCon_CC(i),..., IndCon_CC(i+1) -1. This calls in for the index of the neighbor cells.

4) compute the interpolant function at ivnb and perform the summation

$$Q_{int} = Q_{int} + d_i * Q(\text{ivnb}) \quad (23)$$

5) compute the summation for weighting function to calculate the interpolant

$$\text{sumdi} = d_i + \text{sumdi} \quad (24)$$

End the loop

6) Evaluate the interpolant function

$$Q_{int} = Q_{int}/\text{sumdi} \quad (25)$$

Parameters compared: power, time step size and pressure under relaxation

Pressure under relaxation:

The under-relaxation of equations, also known as implicit relaxation, is used in the pressure-based solver to stabilize the convergence behavior of the outer nonlinear iterations by introducing selective amounts of φ in the system of discretized equations. This is equivalent to the location-specific time step.

General approach for under relaxation can be given as

$$\varphi_C^{new,used} = \varphi_C^{old} + U(\varphi_C^{new,predicted} - \varphi_C^{old}) \qquad (26)$$

Where, φ_C denotes the variable in current cell. Here U is the relaxation factor:

- U < 1 is under relaxation. This may slow down speed of convergence but increases the stability of the calculation, i.e. it decreases the possibility of divergence or oscillations in the solutions.
- U = 1 corresponds to no relaxation. One uses the predicted value of the variable.
- U > 1 is over relaxation. It can sometimes be used to accelerate convergence but will decrease the stability of the calculation.

3.4.2. MODIFIED SHEPHERD INTERPOLATION

Initialize

1) Determine Euclidean distance at icell_i

$$\mathrm{d} = \left(\mathrm{X_c(icell\_i)} - \mathrm{x_{gint}}\right)^2 + \left(\mathrm{Y_c(icell\_i)} - \mathrm{y_{gint}}\right)^2 + \left(\mathrm{Z_c(icell\_i)} - \mathrm{z_{gint}}\right)^2 \qquad (27)$$

2) To determine the weighting function at icell_i

   As the weighting function takes the below form in MSM,

$$\mathrm{w_i} = \frac{\left[\frac{\mathrm{R}-\mathrm{d_i}}{\mathrm{Rd_i}}\right]^2}{\sum_{\mathrm{j}=1}^{\mathrm{n}}\left[\frac{\mathrm{R}-\mathrm{d_j}}{\mathrm{Rd_j}}\right]^2} \qquad (28)$$

   It's simplified as below, to reduce the computation effort

$$\mathrm{rd} = \sqrt{\mathrm{r} * \mathrm{d}} \qquad (29)$$

$$\mathrm{W} = \frac{(\mathrm{r}+\mathrm{d}-\mathrm{rd}-\mathrm{rd})}{\mathrm{r}*\mathrm{d}} \qquad (30)$$

3) To Determine interpolant function at icell_i

$$Q_{int} = W * Q(icell\_i) \quad (31)$$

4) Initialize summation for weighting function

$$sumW = W \quad (32)$$

Calculation

Start the loop

5) Determine Euclidean distance at distance from each neighbors to the ghost cell

$$d = \left(X_c(ivnb) - x_{gint}\right)^2 + \left(Y_c(ivnb) - y_{gint}\right)^2 + \left(Z_c(ivnb) - z_{gint}\right)^2 \quad (33)$$

where ivnb =IndCon_CC(i),..., IndCon_CC(i+1) -1. This calls in for the index of the neighbor cells.

6) Same steps as in initialization is repeated for to determine the weighting function, but only if $(d < r)$

$$rd = \sqrt{r * d} \quad (34)$$

$$W = \frac{(r+d-rd-rd)}{r*d} \quad (35)$$

7) compute the interpolant function at ivnb and perform the summation

$$Q_{int} = Q_{int} + W * Q(ivnb) \quad (36)$$

8) compute the summation for weighting function to calculate the interpolant
$sumW = sumW + W$(37)

End the loop

9) Evaluate the interpolant function

$$Q_{int} = Q_{int}/sumW \quad (38)$$

Compared for different values of r

3.4.3. ARTIFICIAL NEURAL NETWORK WITH BASIS FUNCTION

Initialization:

1) Determine a number *n* of basis functions of the network structure
2) Allocate the index of neighbor cells to an array Iadr,, such that the index of Iadr array is indexed from $1,2,\dots,n$ which facilitates to calculate the weight functions, which the Euclidean distance between all the neighbor cells, rather from the current cell icell_i

Calculation:

3) Calculate the interpolation matrix A(I,j) which is $\varphi_j(x_i)$
4) Inverse the matrix using Generalized Inverse matrix method. Pseudo Moore approach is followed. G is the generalized matrix, such that

$$AGA = A \tag{39}$$

This is a special case, and classical inverse matrix cannot be followed, as it's a matrix of Euclidean distance for which diagonal elements are zero. A real symmetric matrix $n \times n$ matrix A is called Euclidean distance matrix (EDM) if there exist points $p_1, p_2, \dots, p_n \in R^k$ such that

$$d_{ij} = (p_i - p_j)'(p_i - p_j), i, j = 1,2, \dots, n \tag{40}$$

With Euclidean distance matrix, when $\mathrm{i} = \mathrm{j}$, the matrix become non invertible, which has to be resolved.

This is done by a subroutine Ginv, which is called inside the calculation. Here A is an array which is given as an input and the returned inverse is also A.

5) Initialize Q_{int}

6) Evaluate the vector

$$W_i = Q(Idr(j) * A(i,j)) \text{ where, } i = 1,2, \dots n \text{ and } j = 1,2, \dots n \tag{41}$$

7) Estimate the interpolant function,

$$Q_i = W_i * \varphi_i(\mathrm{x_{gint}}) \text{ where, } i = 1,2, \dots n \tag{42}$$

8) Perform the summation

$$Q_{int} = Q_{int} + Q_i \tag{43}$$

CHAPTER IV RESULTS AND DISCUSSION

4.1. INVERSE DISTANCE WEIGHT FUNCTION

Inverse distance weight function which also otherwise called as Shepard Interpolation function is a simplest of all point scattered data interpolation which uses deterministic approach to evaluate the interpolated function from the nearest points. To begin with the evaluation of various interpolation functions, it was considered wise to start with a simple approach and to proceed further with complex functions. Since no bench work existed in this field, logical reasoning was used to compare the results between different approaches tried.

Inverse distance weighted (IDW) interpolation explicitly implements the assumption that things that are close to one another are more alike than those that are farther apart. To predict a value for any unmeasured location, IDW uses the measured values surrounding the prediction location. The measured values closest to the prediction location have more influence on the predicted value than those farther away. IDW assumes that each measured point has a local influence that diminishes with distance. It gives greater weights to points closest to the prediction location, and the weights diminish as a function of distance, hence the name **inverse distance weighted.**

This is implemented in the code using a simple loop by calling the neighbor indexes from IndCON_CC array (See section 3.4.1).

GRIDS WITHOUT INTERACTIONS:

In the initial stage when the icell_e was not included in the routine to interpolate the values, there were no interactions found between the grid interfaces. This is shown in the Figure 14. Also the symmetric vortex that forms behind the moving grid space due to the incontinuity in the fluid flow was an erroneous result which was corrected in the further calculations.

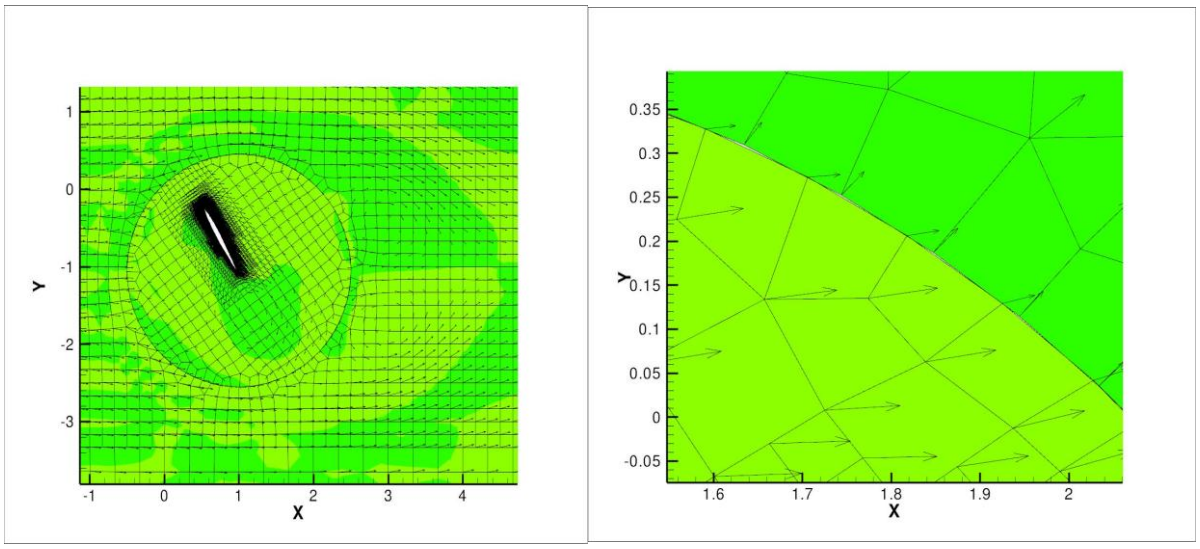


Figure 14: Sliding Grid without interactions

This was corrected by calling the reference of the external cell to interpolate the value between the internal and external cells. This ensured a better communication at the interface. This is shown in the Figure 15.

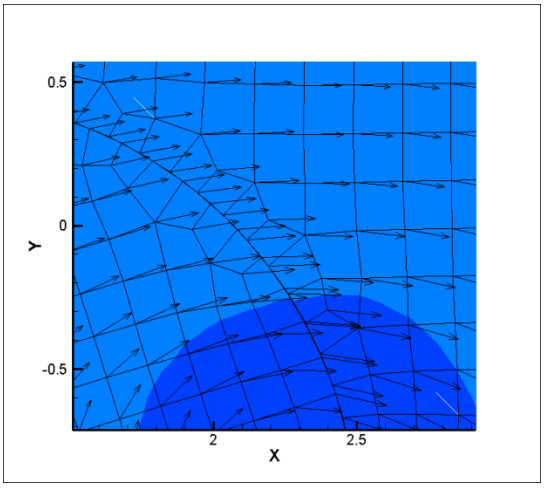


Figure 15: Sliding Grid with interactions

The results are compared based on three parameters,

1. With different power functions
2. With varying time step
3. With reducing the pressure under-relaxation

4.1.1. WITH DIFFERENT POWER FUNCTIONS

As mentioned above, weights are proportional to the inverse of the distance (between the data point and the prediction location) raised to the power value *p*. As a result, as the distance increases, the weights decrease rapidly. The rate at which the weights decrease is dependent on the value of *p*. If *p* = 0, there is no decrease with distance, and because each weight w_i is the same, the prediction will be the mean of all the data values in the search neighborhood. As *p* increases, the weights for distant points decrease rapidly. If the *p* value is very high, only the immediate surrounding points will influence the prediction.

Hence the results are compared with different values of p to find how significant the changes are when the dependency of the neighbors are reduced or increased.

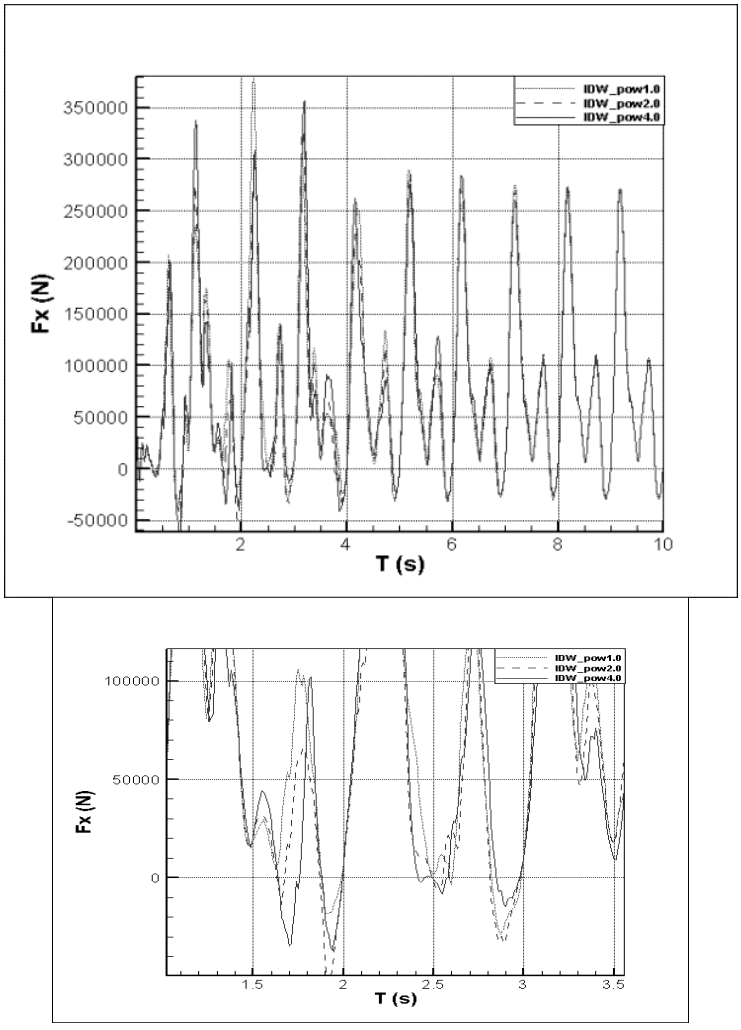


Figure 16:IDW Force history in x direction for timestep size 0.01

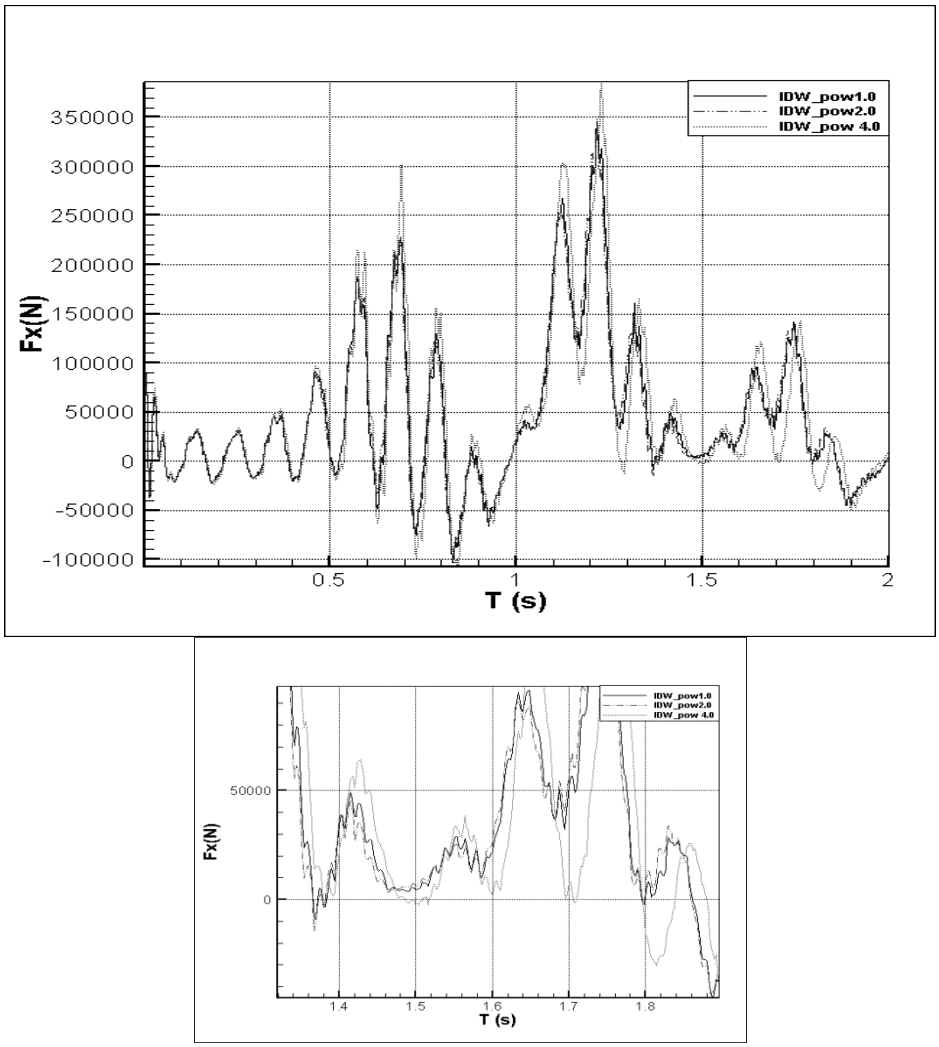
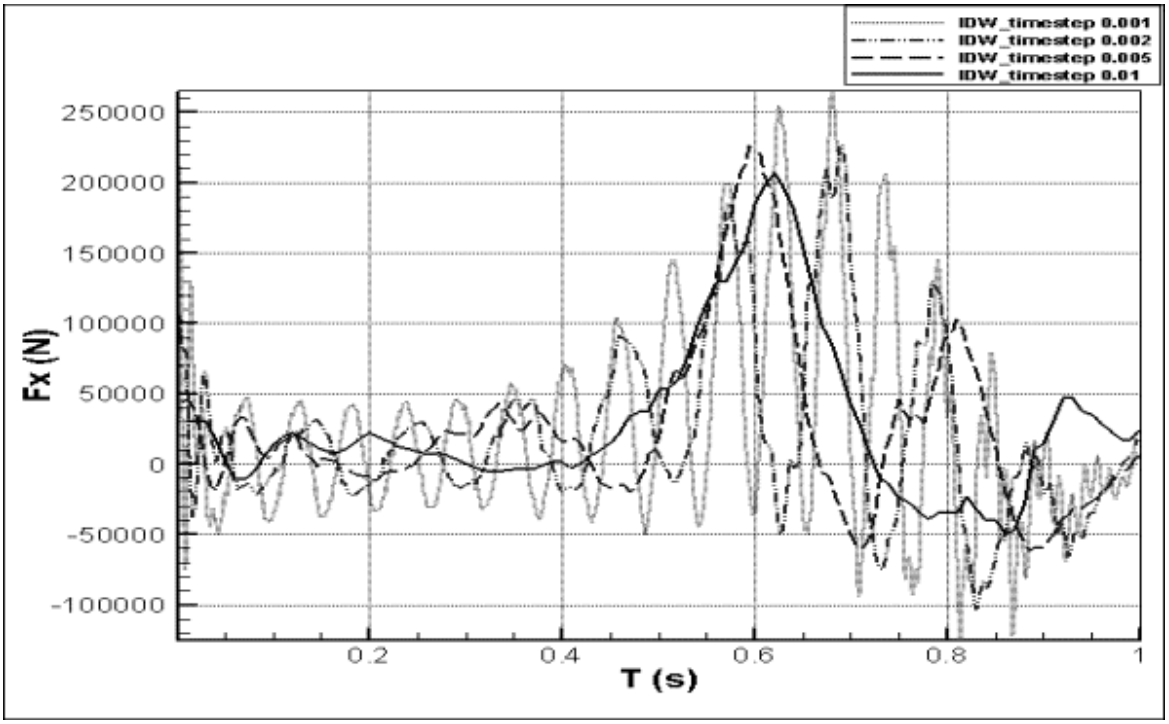


Figure 17:IDW Force history in x direction for timestep size 0.002

From the Figure 16 with the timestep size of 0.01, with the power value of 2.0, its observed that the force variation with the rotating motion of the aerofoil is well captured, while the power 4.0 gives sharp variation and instabilities observed with power 1.0. From the Figure 17, with the fine timestep size of 0.002, though there are instabilities observed with all the three diferent power functions, the solution is significantly better with power 2.0. Hence with the close observation with all the plots above show that the power 2.0 is considerably stable with other two power values. All it should be noted that for the fine timestep size, significant difference among the three different power functions are observed at the later stage when the flow is highly turbulent, but for the timestep size of 0.01, significant difference observed initially diminishes in the later stage.

4.1.2. WITH VARYING TIMESTEP

Interpolation with different timestep size is compared to analyze the influence of timestep size in the accuracy of the solution. Different timestep size of 0.001 to 0.01 is chosen.



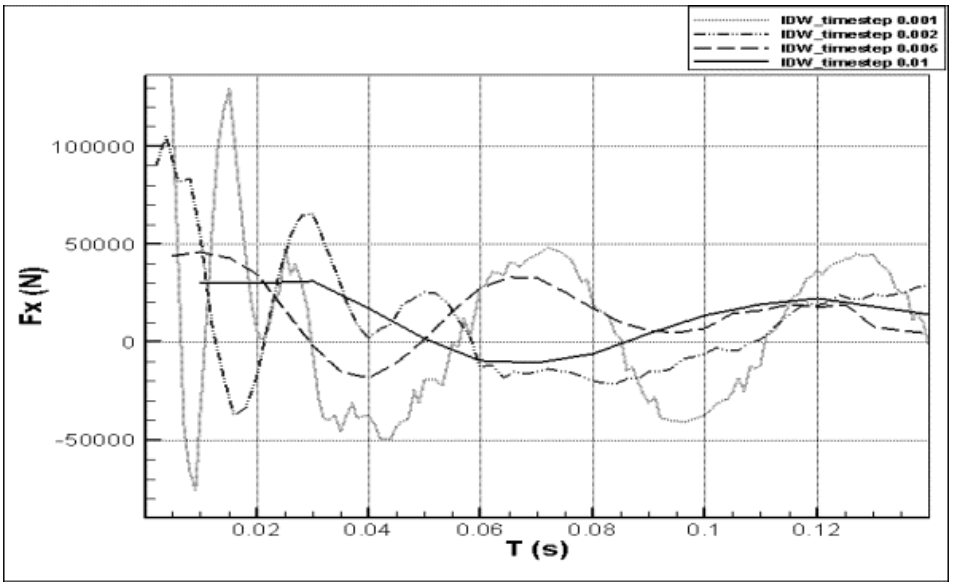


Figure 18: IDW Force history in x direction for power 1.0

Its shows with the finer timestep size, the drag force are well captured from Figure 18. Though with a very fine timestep size of 0.001s there are some oscillations observed in the plot, due to the numerical instability of the solution. This is further analyzed by tuning the pressure under relaxation parameter in the following stages. As we could also observe, large timestep size of 0.01 doesn't give better results as when comparing with finer one.

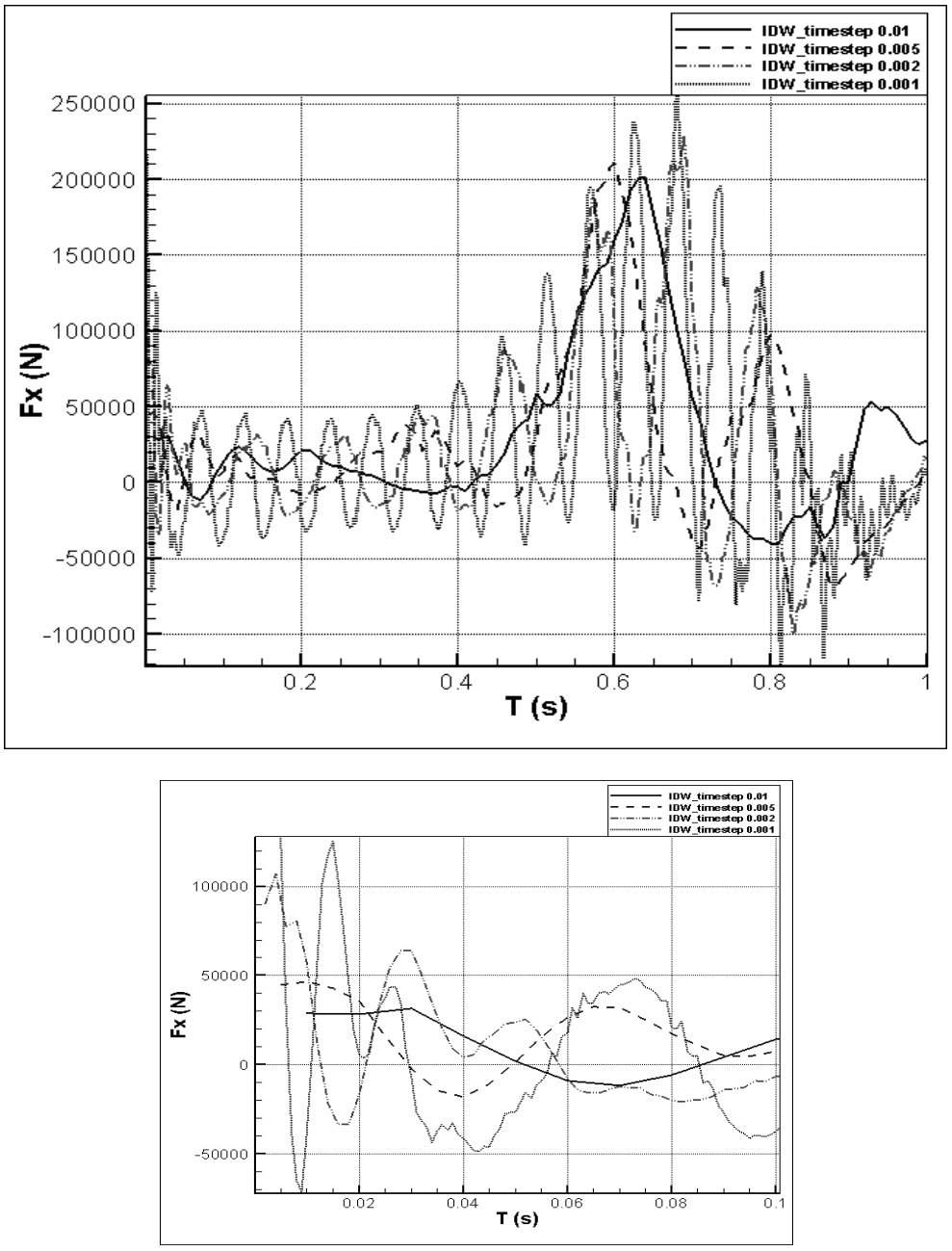


Figure 19: IDW Force history in x direction for power 2.0

Plots shows similar observations as with power 1.0. So bigger power of 4.0 is selected to evaluate further.

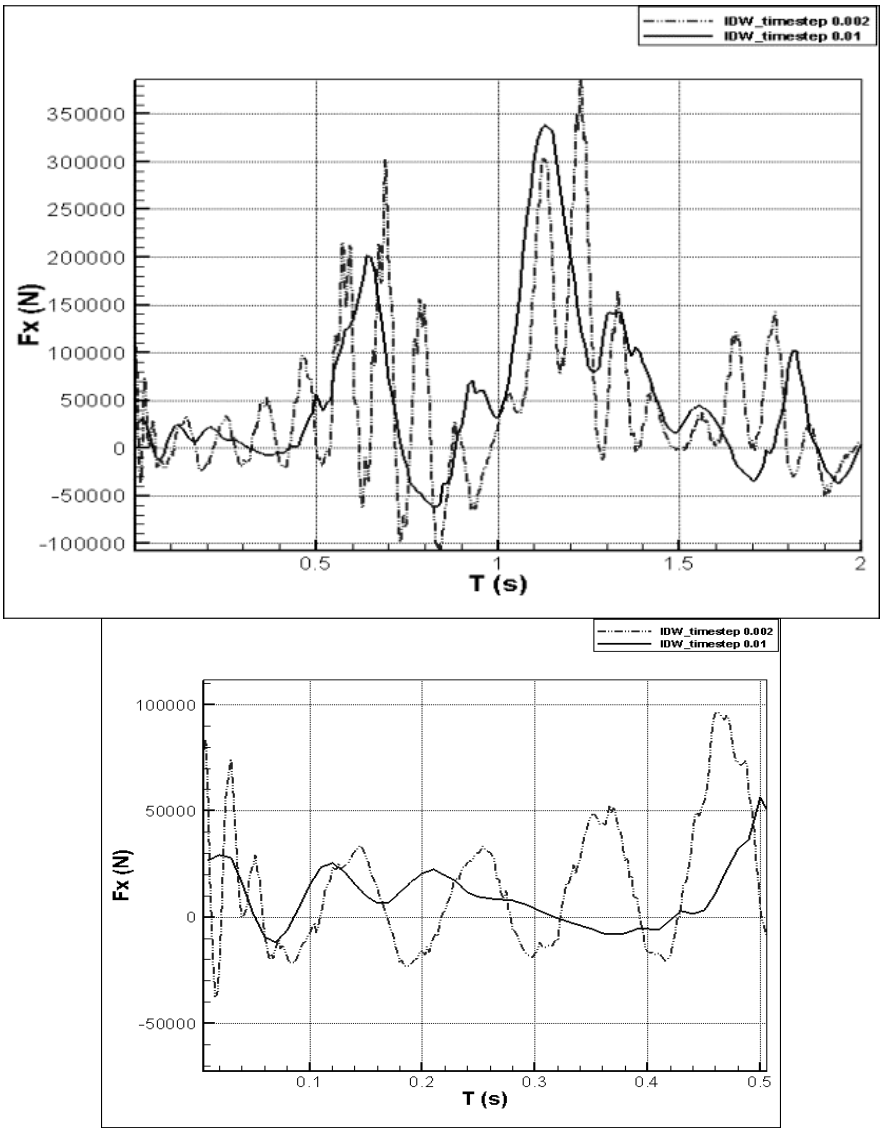


Figure 20: IDW Force history in x direction for power 4.0

With the power 4.0, for timestep 0.002 there appears to be less noise in the solution initially. This instability in the initial stage was considered to be an issue which has to be resolved. But even then, with power 4.0, in the later stage, it doesn't capture the drag force as comparing with other two cases. The plot (Figure 20) shows sharp interfaces and high instabilities even with timestep size 0.002

Though the results from power 1.0 is closer to poser 2.0 solutions (Figure 18 and Figure 19), closer observations of the plots shows that the power 2.0 plots are with less oscillations or instabilities in the peaks in the initial stage.

Contour plots:

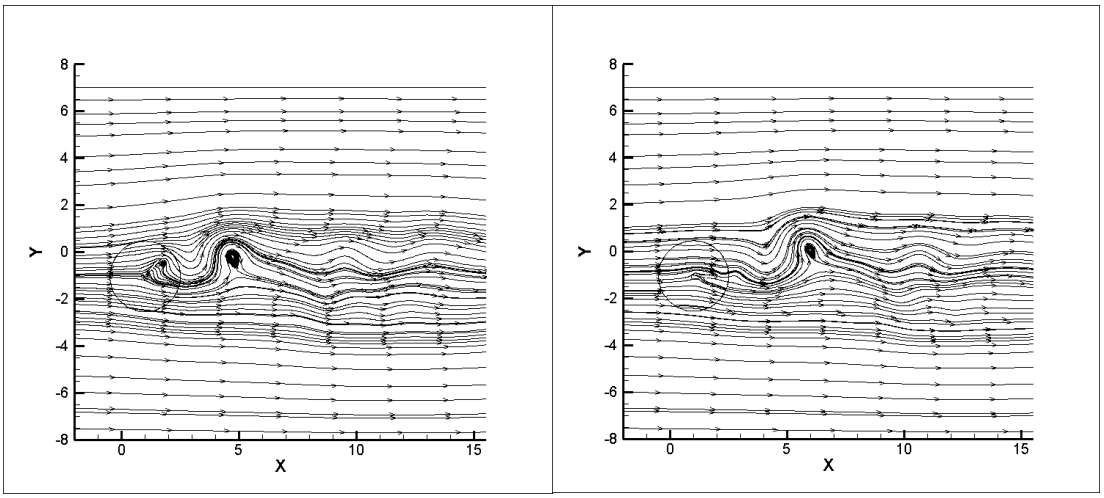


Figure 21: IDW stream line plot at 1.85s and at at 2s for 0.002 timestep size and for power 1.0

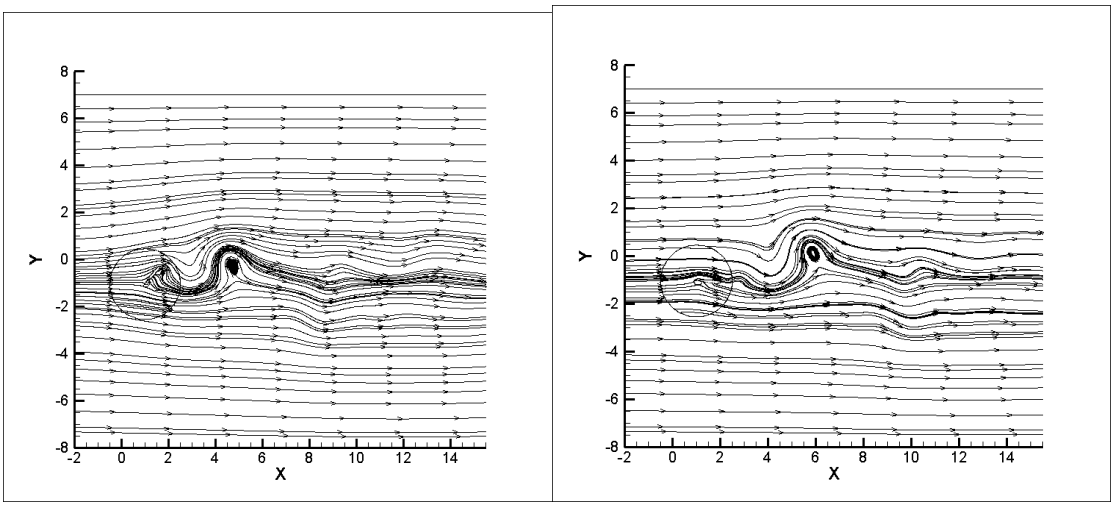


Figure 22: IDW stream line plot at 1.85s and at at 2s for 0.002 timestep size and for power 2.0

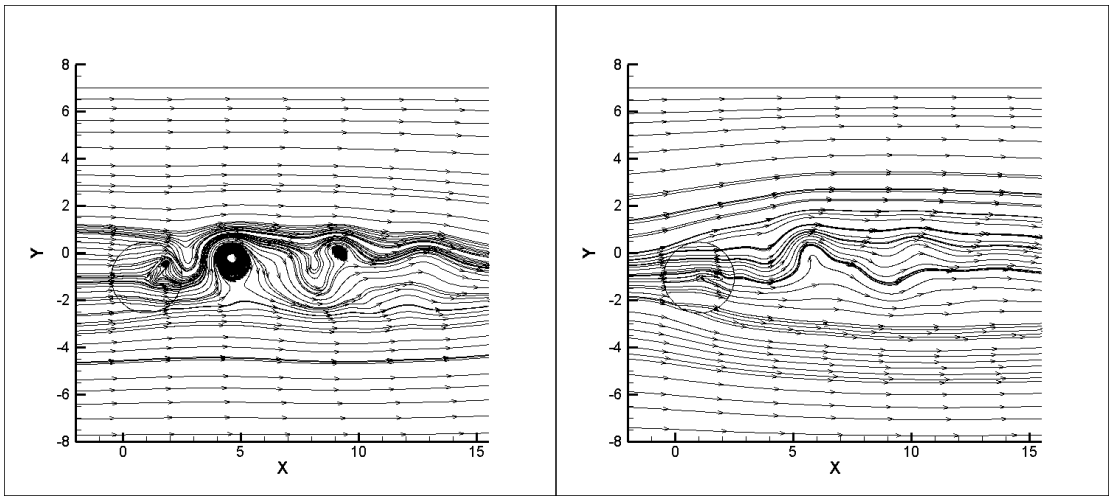


Figure 23: IDW s tream line plot at 1.85s and at at 2s for 0.002 timestep size and for power 4.0

Since there wasn't any significance conclusion can be drawn from the above plots, streamline plots were drawn which did show a significant difference at the trail side of the

moving grid domain at the final stage. As observed in Force plots, there wasn't much difference observed in power 1.0 and power 2.0 plots. But as with power 4.0, the noise at the trail edge due to the turbulence in flow created by the rotating airfoil is well captured. But still due to the sharp variation observed in the Force plots, its found optimum to use power 2.0.

4.1.3. WITH REDUCING PRESSURE UNDER RELAXATION

From the above study with varying power values, as power 2.0 seems to give comparatively stable results, it's further evaluated by reducing the pressure under relaxation between 0.3 to 0.1.

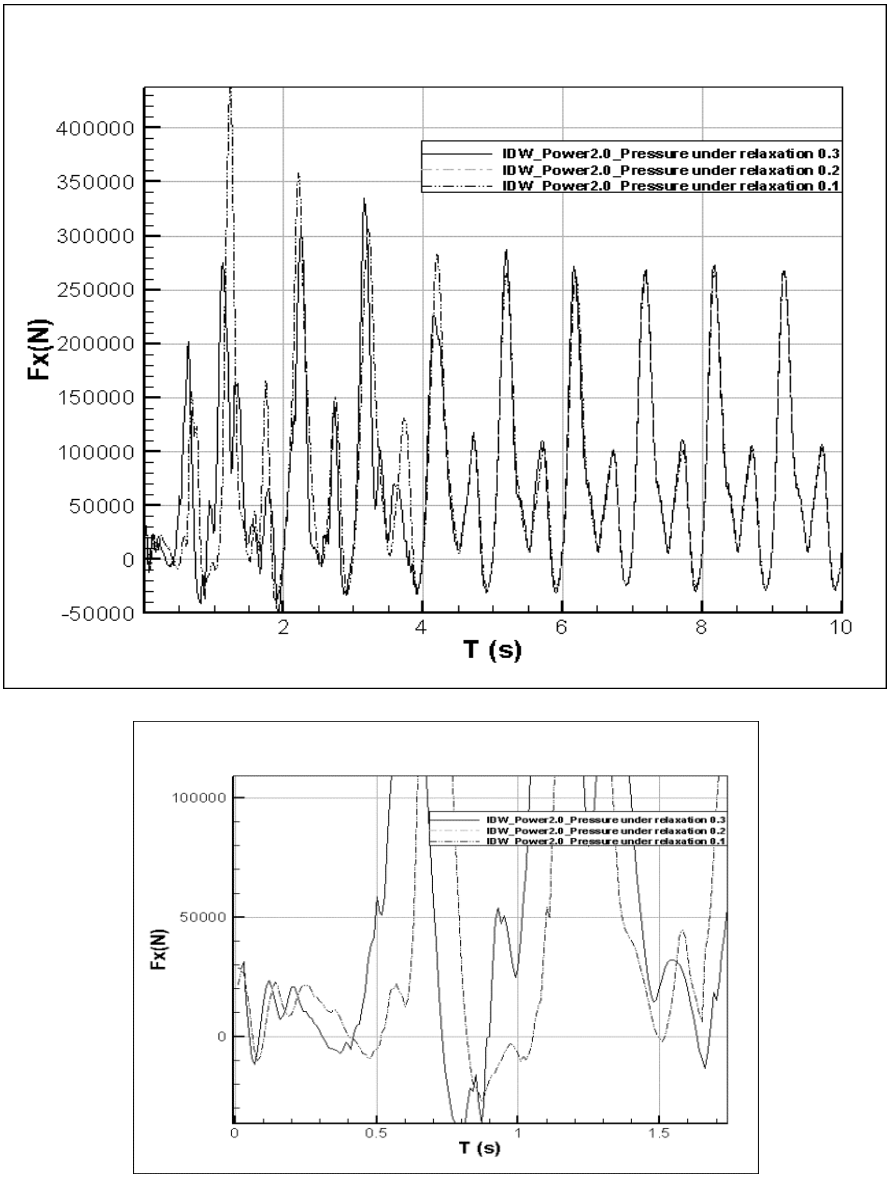


Figure 24: IDW with tuning pressure under-relaxation between 0.3 and 0.1 for power 2.0 and timestep 0.01

By reducing the under pressure relaxation oscillations tend to increase in the initial state. Under relaxation parameters are defined to reduce these oscillations (Figure 24).

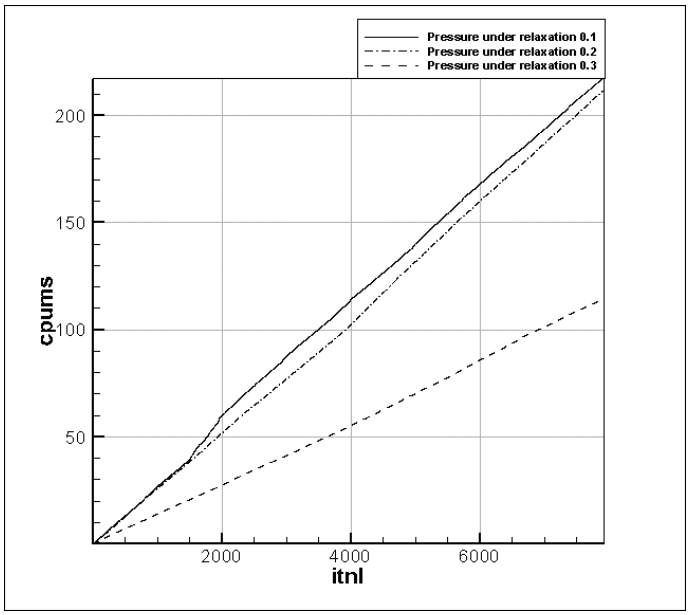


Figure 25: Convergence plot for pressure under-relaxation for power 2.0 and timestep 0.01

Also from the Figure 25, it's clear that the pressure under relaxation 0.3 converges fast when compared to 0.1 and 0.2 due to the stability that pressure under relaxation offers. From this it can be inferred that minimum of 0.3 pressure under relaxation can be given to increase the convergence rate by offering a better stability. But it should be noted many parameters contribute to the convergence rate along with this.

4.2. MODIFIED SHEPARD'S METHOD

Modified Shepard algorithm is modified classical Inverse distance weighted approach, which uses radial domain to select the n set of neighbor data. Due to this reason, though the communication between the interfaces is well defined, there wasn't any significant influence of r value in the Figure 26.

Even with the streamlines that defines the flow, there wasn't any significant difference observed by varying the R value. Obviously, since the domain for the interpolation being no radial, this is an expected observation. But the radial function can be replaced by polynomial functions to get satisfactory results in the future.

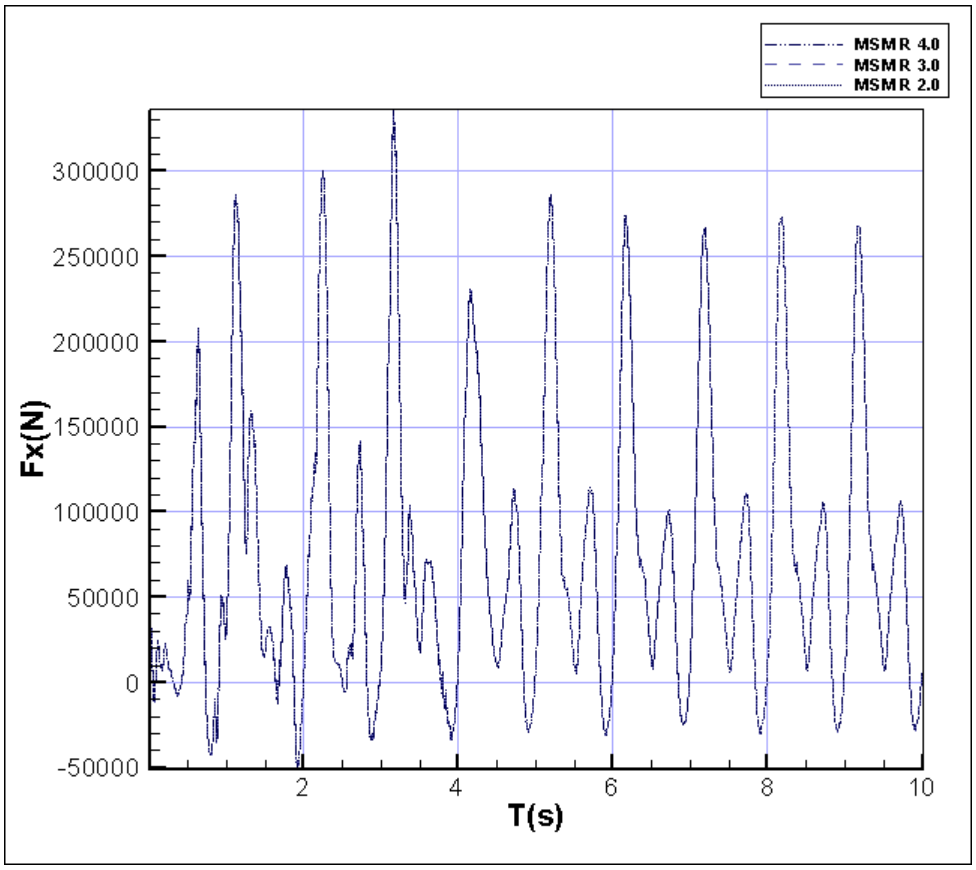


Figure 26: Force plot for Modified Shepard method with different R values

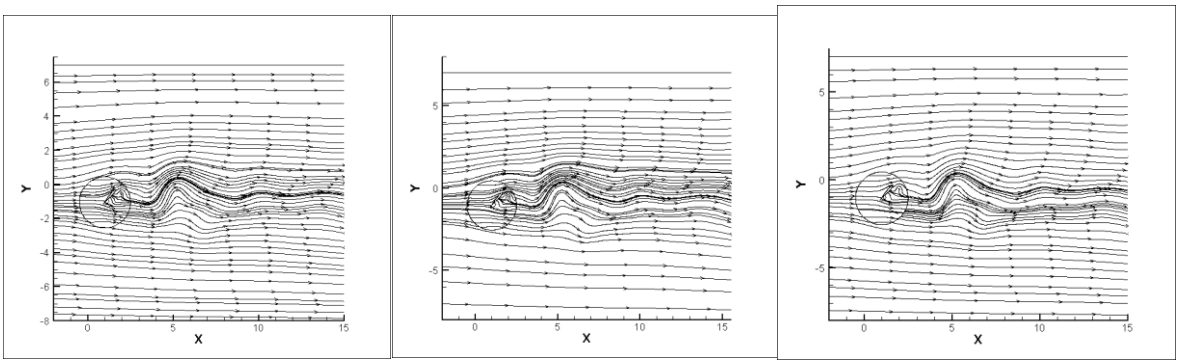


Figure 27: Contour plot Modified Shepard method with different R values 2.0, 3.0 and 4.0 at 4.85s

4.3. ARTIFICIAL NEURAL NETWORK WITH MULTIQUADRATIC FUNCTION

With the multiquadratic function, different parameters like influence of delta function and pressure under relaxation are compared.

4.3.1. WITH DELTA FUNCTION

With small increase in delta function the calculation becomes highly sensitive and initially there is a high instability observed which later becomes stabilized.

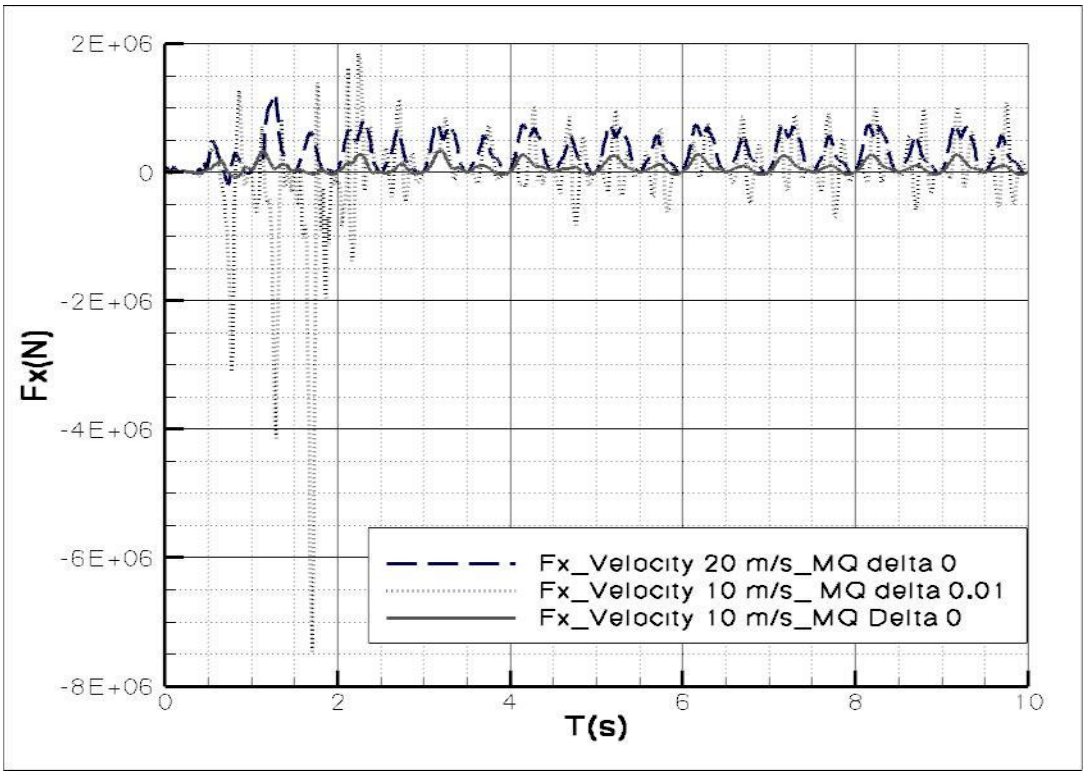


Figure 28: Sensitivity of delta function comparison plot for Multiquadratic interpolation

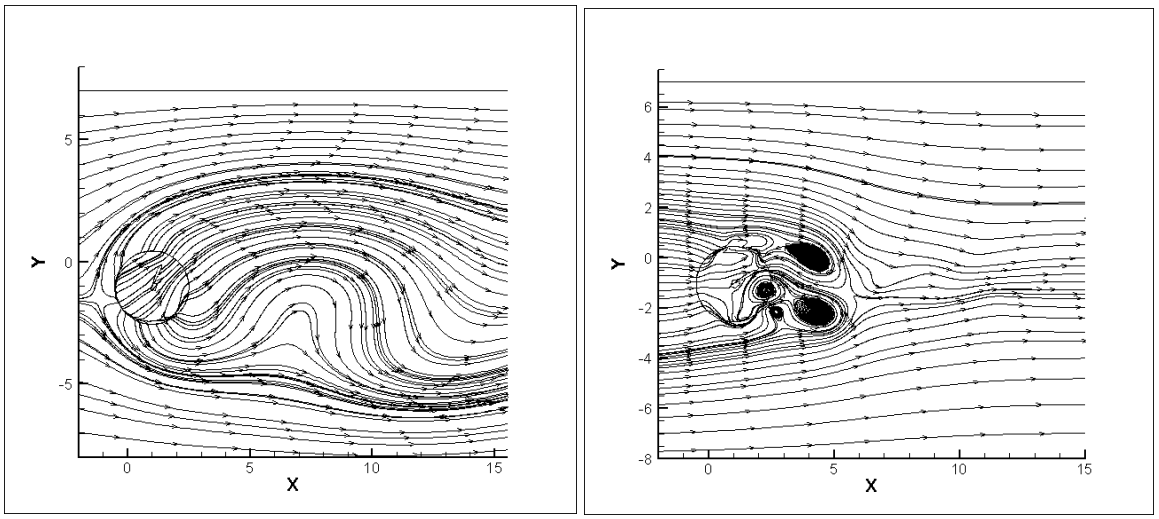


Figure 29: Contour plot for Multiquadratic and Quadratic interpolation with delta function 0.01

Though there was a well-defined communication at the interface, in the later stage the flow behaves as if there is no communication. But with Quadratic function, gives a normal communication at the leading edge, but vortexes at the trailing edge, which can be inferred due to the hindrance to the flow.

4.3.2. WITH REDUCED PRESSURE UNDER RELAXATION

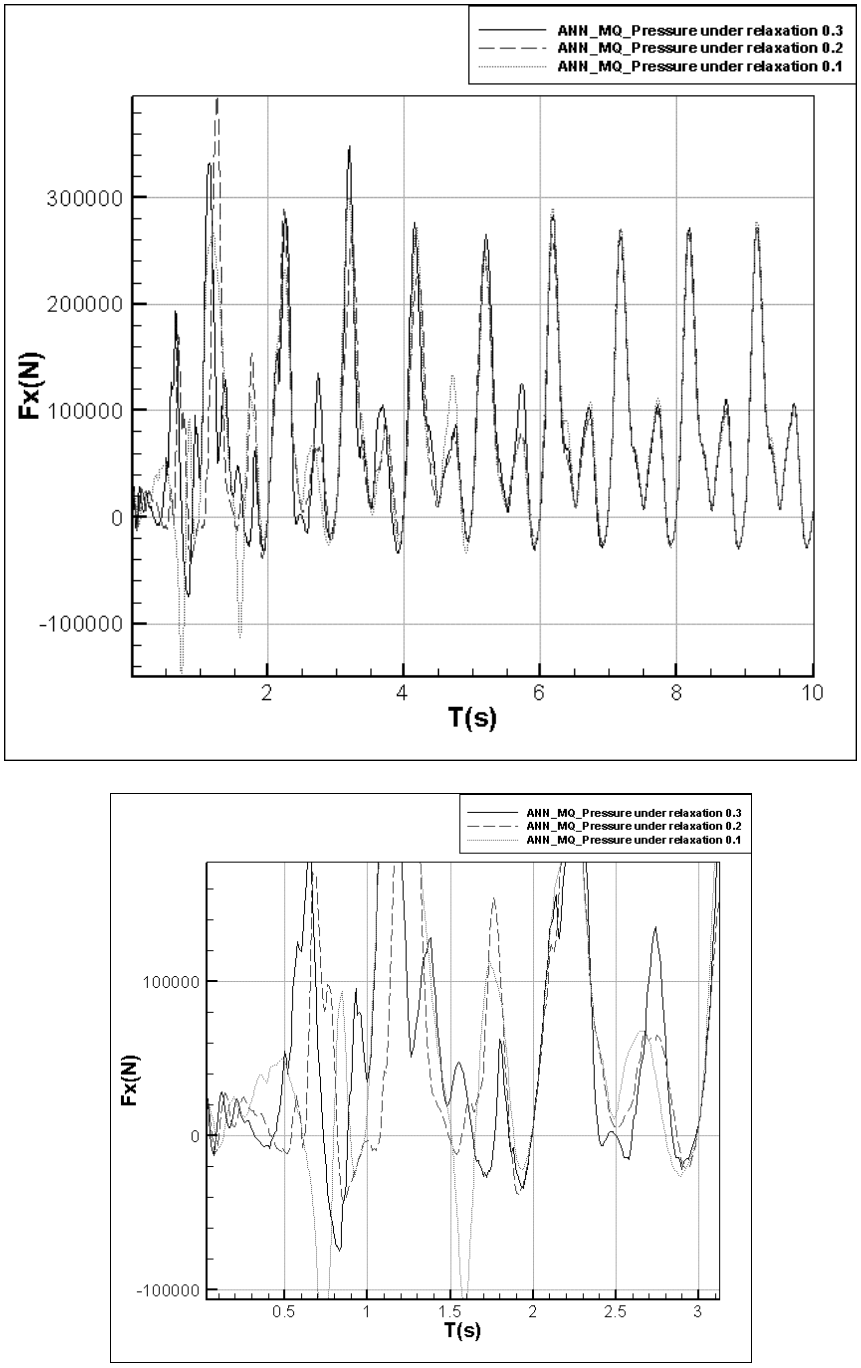


Figure 30: ANN MQ with reduced Pressure under relaxation

Similar to the previous observations, the instability in the beginning of the flow was increased when the pressure under relaxation was reduced. Hence its good to ensure the

optimum pressure under relaxation factor to get better compromise between computation time and accuracy.

4.4. ARTIFICIAL NEURAL NETWORK WITH QUADRATIC FUNCTION

With the multiquadratic function there was a significant difference in the delta function, which was not the case with Quadratic function. The solution was comparatively stable with 0.01 delta function (Figure 31)

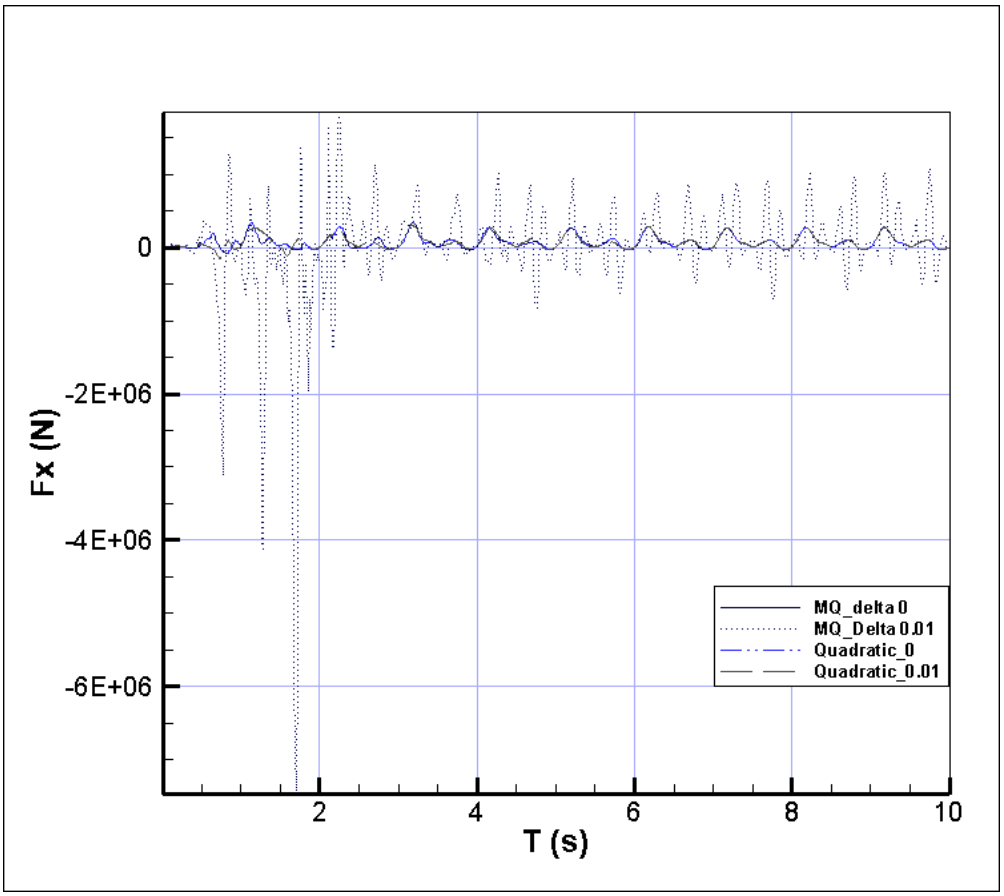
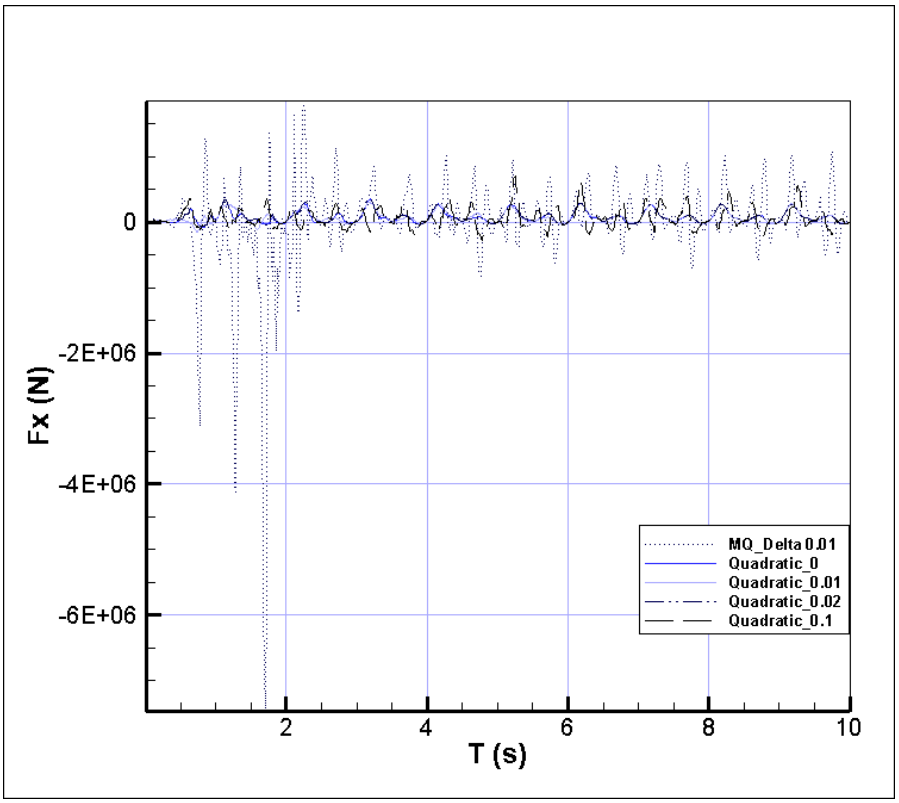


Figure 31: Multiquadratic vs Quadratic

As with contour plots also, this difference was clearly seen. As in the Quadratic function, the communication was not completely lost with the time, which was not the case in Multiquadratic. Further to this Quadratic function method was evaluated with increasing delta function. By increasing the delta function, the stability was lost (Figure 32), but not as sensitive as in case of Multiquadratic approach.



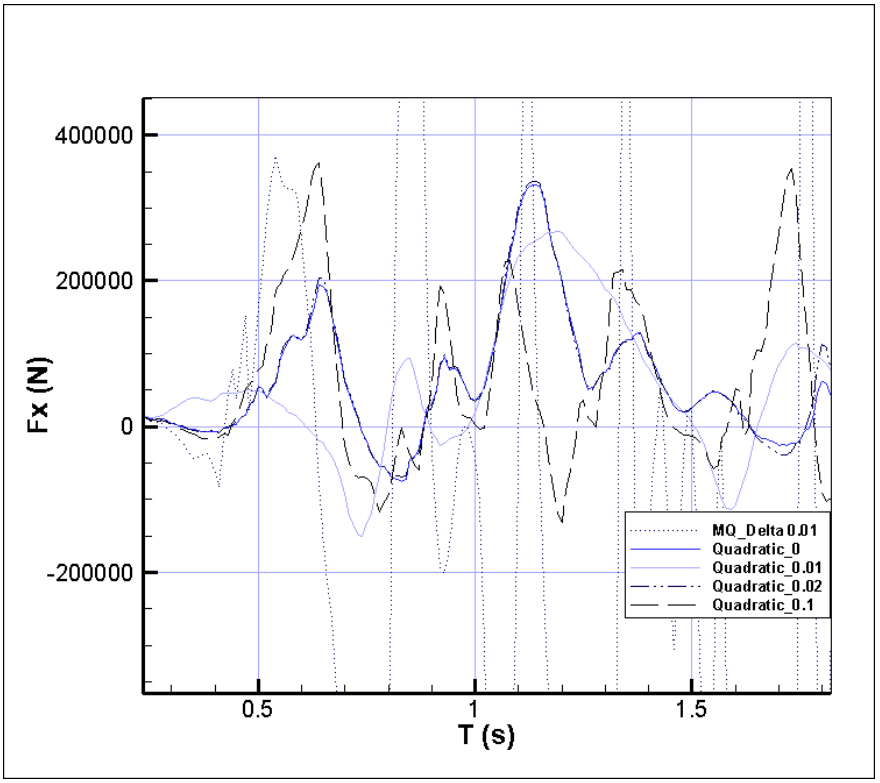


Figure 32: Quadratic function with different delta value

4.5. COMPARISON OF ARTIFICIAL NEURAL NETWORK WITH MSM AND IDW

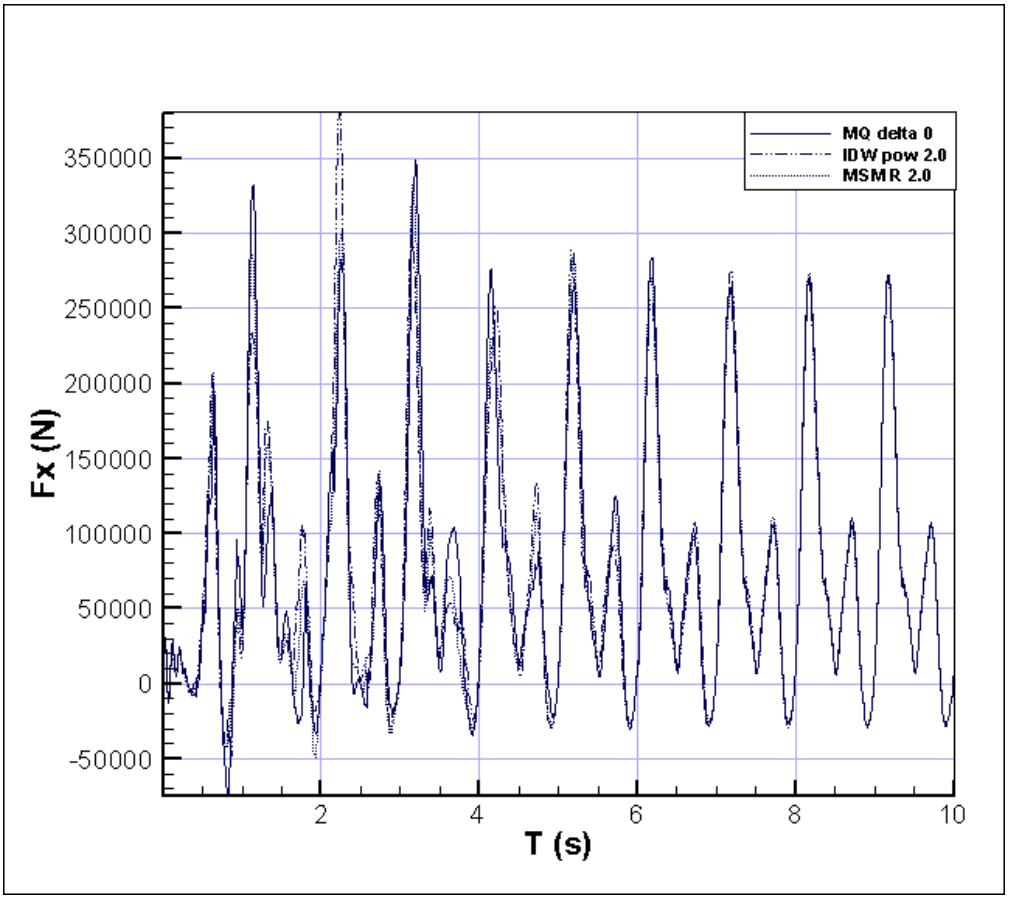


Figure 33: Comparison of all three methods ANN, MQ and IDW

From the Figure 33, it was observed that both MQ and IDW has notable gradients when the flow starts and then become stabilized. ANN multiquadratic function has negative peaks and IDW has positive peaks. But MSM was observed with a stable flow. As the MSM method doesn't seems be logically applicable to our application, much investigation is required on R function.

CHAPTER V CONCLUSION AND FUTURE PROSPECTIVE

Different ways of communication at sliding grid interface for moving grid domain was discussed and analyzed. Three different approaches, Inverse distance weighted average method, Modified Shepard Algorithm and Artificial Neural network with Multiquadratic and Quadratic basis function are compared and studied.

Inverse distance method was studied explicitly and it shows better results with fine time step and with pressure under relaxation of 0.3. Also power 4.0 show better results in streamline plots. If there are more neighbor cells which is possible with fine mesh, one can expect good correlation with power 2.0 also. But, since the mesh size was fixed, this behavior can be well studied by having different mesh sizes, and hence the influence of the power function on the solution can be understood. Also Inverse distance weighting function does not require solving linear system of equations as in ANN. With its simple algorithm and similar accuracy with other two methods, Inverse distance method is the most preferred one However Point-by-point methods do not rely on connectivity information as they determine the displacement of each point in the flow mesh based on its relative position with respect to the domain boundary. And as it can't be further developed as it's a classical approach, this limits its application extensively.

This calls for the further development of other two approaches. Modified Shepard method with its simple structure gives more stable results. Only disadvantage of this method is that, it can't be logically suitable for our application as it uses radial domain for selecting n set of data points. But by analyzing with different functions like, polynomial based or root mean square method based functions, better results can be expected.

As it was referred in the literature, artificial neural network seems to give more accurate predictions for Geostatistics and medical image processing industry. But as far as our application, it doesn't seem to give accurate prediction. But with a very robust algorithm for inverse matrix or with more detailed definition for delta function, this method can give closer results. Also within the time frame of our work, as it was not possible to validate with the real test case, it's possible to get better comparison of all three methods with help of 3D experiments.

APPENDIX

Appendix 1: FORTRAN 90 code for Inverse distance weighting

```
Subroutine Interp_IG (imesg, IpntCF_CC, IndCon_CC, Xc, Yc, Zc, Q, icell_i,
icell_e, &
     xgint, ygint, zgint, Qint)

  Implicit None

  ! in
  Integer, Dimension(*), Intent(in) :: IpntCF_CC, IndCon_CC
  Integer, Intent(in) ::  imesg, icell_i, icell_e
  Double Precision, Dimension(*), Intent(in) :: Q, Xc, Yc, Zc
  Double Precision :: xgint, ygint, zgint
  !
  ! out
  Double Precision, Intent(out) :: Qint
  !
  ! local
  Integer ::  ipvnb, ivnb
  Double Precision :: di, sumdi, pow
  !
  !
  pow = 1.0
  !  - Initialisation
  di = 1.0/Max(1.e-10,Sqrt( (Xc(icell_i)-xgint)**2 + (Yc(icell_i)-ygint)**2
+ (Zc(icell_i)-zgint)**2 ))**pow
  Qint = di*Q(icell_i)
  sumdi = di
  !  - Summation of neighbour cells
  Do ipvnb = ipntCF_CC(icell_i)+1, ipntCF_CC(icell_i+1) - 1
     ivnb = IndCon_CC(ipvnb)
        If ( ivnb == icell_e ) Cycle
                di = 1.0/Max(1.e-10,Sqrt( (Xc(ivnb)-xgint)**2 + (Yc(ivnb)-
ygint)**2 + (Zc(ivnb)-zgint)**2 ))**pow
                Qint = Qint + di*Q(ivnb)
                sumdi = sumdi + di
  End Do
  Qint = Qint/sumdi
  ! write(imesg,*) 'interp_ig: ',Qint  !test
  !
End Subroutine Interp_IG
```

Appendix 2: FORTRAN 90 code for Modified Shepard method

```
Subroutine Interp_IG (imesg, IpntCF_CC, IndCon_CC, Xc, Yc, Zc, Q,
icell_i, icell_e, &
     xgint, ygint, zgint, Qint)

  Implicit None

  ! in
  Integer, Dimension(*), Intent(in) :: IpntCF_CC, IndCon_CC
  Integer, Intent(in) ::  imesg, icell_i, icell_e
  Double Precision, Dimension(*), Intent(in) :: Q, Xc, Yc, Zc
  Double Precision :: xgint, ygint, zgint

  ! out
  Double Precision, Intent(out) :: Qint

  ! local
  Integer ::  ipvnb, ivnb
  Double Precision :: d, sumW, r, W, rd

! Modified Sheperd Method Interpolation : Qint = W(1)*Q(1) + W(2)*Q(2)
+-------+W(N)*Q(N) / W(1)+W(2)+---------W(N)
! where W(K) (x,y,z) = ( (R(K)-D(K))+ / R(K)*D(K) )**2
! where (R(K)-D(K))+ = 0 If R(K) < D(K)
! and D(K) is the Euclidean distance between (x, y, z) and Qint
! where R is the radius of influence specified by the user.
! let r = R**2
! let d = D**2
!
  !  - Initialisation
  r = 4
  d = (Xc(icell_i)-xgint)**2 + (Yc(icell_i)-ygint)**2 + (Zc(icell_i)-
zgint)**2
  rd = sqrt ( r * d)
  W = ( r + d - rd - rd ) / ( r * d)
  Qint = W * Q(icell_i)
  sumW = W
    !  - Summation of neighbour cells
  Do ipvnb = ipntCF_CC(icell_i)+1, ipntCF_CC(icell_i+1) - 1
     ivnb = IndCon_CC(ipvnb)
     If ( ivnb == icell_e ) Cycle
        If (d < r ) then
                d = (Xc(ivnb)-xgint)**2 + (Yc(ivnb)-ygint)**2 + (Zc(ivnb)-
zgint)**2
                rd = sqrt ( r * d)
                W = ( r + d - rd - rd ) / ( r * d)
                Qint = Qint + W*Q(ivnb)
                sumW = sumW + W
        end if
  End Do
  Qint = Qint/sumW
    ! write(imesg,*) 'interp_ig: ',Qint !test
!
End Subroutine Interp_IG
```

Appendix 3: FORTRAN 90 code for Artificial neural network basis function Interpolation

```
Subroutine Interp_IG (imesg, IpntCF_CC, IndCon_CC, Xc, Yc, Zc, Q,
icell_i, icell_e, &
     xgint, ygint, zgint, Qint)

  Implicit None

  ! in
  Integer, Dimension(*), Intent(in) :: IpntCF_CC, IndCon_CC
  Integer, Intent(in) ::  imesg, icell_i, icell_e
  Double Precision, Dimension(*), Intent(in) :: Q, Xc, Yc, Zc
  Double Precision :: xgint, ygint, zgint

  ! out
  Double Precision, Intent(out) :: Qint

  ! local (Artificial neutral network(ANN) approach with Quadratic
Function)

  Integer ::  ipvnb, ivnb, ipvnb_inner, ivnb_inner
  Double Precision :: delta, Qi, Wicelli

! pour inverse matrix

  Integer ::  m, n, i, j, k
  Integer, DIMENSION(:), allocatable :: Iadr        ! TEST correspondane
between "i" and "ivnb"
  real, dimension(:,:), allocatable :: A
  REAL,  DIMENSION(:), allocatable :: W

  !write(imesg,*) '******************************** My lib RBF
*******************************'!!!test
!!!to find the interpolation function Qint by using Artificial neutral
network(ANN) approach
!!!(last modified 06/06/2012_implemented ginv code as per B. Rust).
!!!The euclidean distance matrix is calculated using generalised matrix
inverse approach!!!!!
  !  - Initialisation
!to calculate n
 n=0
  Do ipvnb = ipntCF_CC(icell_i)+1, ipntCF_CC(icell_i+1) - 1
   ivnb = IndCon_CC(ipvnb)
    If ( ivnb == icell_e ) Cycle
    n=n+1
  End do
 allocate (Iadr(n))
 allocate (A(n,n))
 allocate (W(n))
  Do i = 1,n
   Do ipvnb = ipntCF_CC(icell_i)+1, ipntCF_CC(icell_i+1) - 1
    ivnb = IndCon_CC(ipvnb)
     If ( ivnb == icell_e ) Cycle
      Iadr(i)=ivnb
   End do
 End do
```

ANN Page 1/3

```
!to calculate interpolation matrix
  ! delta = 3 !smoothing function of ANN
  delta = 0.0 !smoothing function of ANN
 Do i =1,n
  Do j =1,n
    A (i,j)  =  (((Xc(Iadr(i))-Xc(Iadr(j)))**2 + (Yc(Iadr(i))-Yc(Iadr(j)))
**2 + (Zc(Iadr(i))-Zc(Iadr(j)))**2 ) + delta**2)
  End do
 End do
!to calculate interpolation matrix, call the subroutine GINV
 call GINV(A)
!
!to calculate the weights
!initialization
 Qint =Q(icell_i)
  Do i = 1,n
    W (i)=0
     Do j = 1,n
      W (i) =  W (i)+ (Q(Iadr(j))*A(i,j))
     End do
  End do
  Do i = 1,n
    Qi = W(i) * (((Xc(Iadr(i))-xgint)**2 + (Yc(Iadr(i))-ygint)**2 + (Zc
(Iadr(k))-zgint)**2 )+delta**2)
    Qint = Qint+Qi
  End do
End Subroutine Interp_IG

Subroutine GINV (A,U,JC,KC,AFLAG,ATEMP,N)

! A Simple Algorithm for Computing the Generalized Inverse of a Matrix
!  by B. Rust, W. R. Burrus and C. Schneeberger
!  CACM 9(5):381-387 (May, 1966)
!
! This routine calculstes the Generalized Inverse of input matrix, A,
! and stores the transpose of it in matrix, A.
!
  integer, parameter :: dp = selected_real_kind(15, 307)
  real(kind=DP), dimension(:,:), intent(in out) :: A
  real(kind=DP), dimension(:,:), intent(out) :: U ! U  -> a bookkeeping
matrix.
  real(kind=DP), dimension(:), intent(out) :: AFLAG, ATEMP ! AFLAG and
ATEMP are temporary working vectors
  real(kind=DP) :: FAC, TOL, DOT1, DOT2, PROD
  integer, intent(in) :: N,JC,KC
  integer  :: I,J,K,L,JM1

!  Computes the inner product of columns JC and KC
!  of matrix, A.

      PROD=0.0
      DO I = 1,N
        PROD = PROD + A(I,JC)*A(I,KC)
      END DO
      RETURN
      DO  I = 1,N
        DO J = 1,N
         U (I,J) = 0.0
        END DO
         U(I,I) = 1.0
      END DO                                                ANN Page 2/3
```

```
      FAC = DOT(N,A,1,1)
      FAC= 1.0_DP/SQRT(FAC)
      DO I = 1,N
        A(I,1) = A(I,1) * FAC
      END DO
      DO I = 1,N
        U (I,1) = U(I,1)*FAC
      END DO
      AFLAG(1) = 1.0
!
! Dependent column tolerance, TOL
!
      TOL=10.0*EPSILON(FAC)
      DO J = 2,N
        DOT1 = DOT(N,A,J,J)
        JM1=J-1
         DO L=1,2
          DO K=1,JM1
            ATEMP(K) = DOT(N,A,J,K)
          END DO
          DO K=1,JM1
            DO I = 1,N
              A(I,J) = A(I,J)-ATEMP(K)*A(I,K)*AFLAG(K)
            END DO
            DO I = 1,N
              U(I,J) = U(I,J)-ATEMP(K)*U(I,K)
            END DO
          END DO
         END DO
        DOT2 = DOT(N,A,J,J)
        IF((DOT2/DOT1) <= TOL) THEN
          DO I=1,JM1
            ATEMP (I)=0.0
             DO  K=1,I
              ATEMP(I) = ATEMP(I) + U(K,I)*U(K,J)
             END DO
          END DO
          DO I = 1,N
            A( I,J)=0.0
             DO K=I,JM1
              A(I,J) = A(I,J) - A (I,K)*ATEMP(K)*AFLAG(K)
             END DO
          END DO
          AFLAG(J) = 0.0
          FAC = DOT(N,U,J,J)
          FAC= 1.0_DP/SQRT(FAC)
        ELSE
          AFLAG(J) = 1.0
          FAC=1.0_DP/SQRT(DOT2)
        ENDIF
        DO I = 1,N
          A(I,J) = A(I,J)*FAC
        END DO
        DO I = 1,N
          U(I,J) = U(I,J)*FAC
        END DO
      END DO
      DO J=1,N
        DO I=1,N
         FAC = 0.0
          DO K = J,N
           FAC=FAC+A(I,K)*U(J,K)
          END DO
         A(I,J) = FAC
        END DO
      END DO
      RETURN
End subroutine Ginv                                                    ANN Page 3/3
```

BIBLIOGRAPHY

Arjun Akkala, V. D. (2011,). Development of an ANN Interpolation Scheme for Estimating Missing Development of an ANN Interpolation Scheme for Estimating Missing. *The Open Environmental & Biological Monitoring Journal,*, 4, 21-31.

Ball, Keith. (Oct 21, 1990). Eigenvalues of Euclidean Distance matrices. *Department of Mathematics, Texas A & M University, Texas*.

C Armando Duarte, J. T. (1996). H-p Clouds-An h-p Meshless Method. *Numerical Methods for Partial Differential Equations*, 12, 673-705.

Franke. R., N. G. (1980). Smooth interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering*, 1691–1704.

J. G. Wang, G. R. (2002). A point interpolation meshless method based on radial basis functions. *International Journal for Numerical methods in Engineering*, 54:1623–1648.

J., R. R. (1988). Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software*, 14, 2, 139–148.

J.A.Parker, R. D. (1983). Comparison of interpolating methods for image resampling. *IEEE Trans. Med. Imag., vol.MI-2*, 31-39.

Jin Li, A. D. (2008). *A Review of Spatial Interpolation Methods for Environmental Scientists.* Canberra, ACT 2601, Australia: Geoscience Australia Record 2008/23.

Kansa, E. (1990). *Computers and Mathematics with Applications*, 19:127–161.

Maeland, E. (1988). On the comparison of interpolation methods. *IEEE Trans.Med Image*, vol. MI-7, pp. 213–217.

Martin Beaudoin, H. J. (2008). Development of a generalized grid interface for Turbomachinery simulations with openFOAM. *Open source CFD International Conference 2008.* Berlin, Germany.

Martin Beaudoin, Hrvoje Jasak. (2008). Development of a Generalized Grid Interface for Turbomachinery simulations with OpenFOAM. *Open Source CFD International Conference.* Berlin, Germany.

Most, T. (n.d.). *Approximation of complex nonlinear functions by means of neural networks.* Bauhaus-University Weimar: Institute of Structural Mechanics,.

Nielson, R. Franke and G.M. (1980). Scattered Data Interpolation of Large Sets of Scattered Data. *International Journal of Numerical Methods in Engineering, Vol 15*, 1,691-1,704.

NUMECA International. (n.d.). Retrieved from www.numeca.com.

Queutey P, V. M. (2007). An interface capturing method for free-surface hydrodynamic flows. *Computers & Fluids*, Vol. 36, No. 9, pp.1481-1510.

Thomas M. Lehmann, C. G. (1999). Survey: Interpolation Methods in Medical Image Processing. *IEEE transactions on Medical Imaging*, VOL. 18, NO. 11.

WILLIAM I. THACKER, J. Z. (2009). *Algorithm XXX: SHEPPACK: Modified Shepard Algorithm for Interpolation of Scattered Multivariate Data.* Association for Computing Machinery, Inc.