1-1-2000

Visualization and approximation of post processed computational fluid dynamics data in a virtual environment

Vishant Jude Shahnawaz
Iowa State University

Visualization and approximation of post processed computational fluid dynamics data in a virtual environment

by

Vishant Jude Shahnawaz

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Major Professor: Judy M. Vance

Iowa State University

Ames, Iowa

2000

Graduate College

Iowa State University

This is to certify that the Master's thesis of

Vishant Jude Shahnawaz

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

ABSTRACT

Computational fluid dynamics (CFD) data is obtained as a result of numerical analysis performed on a flow model. In order for this data to be accurately analyzed, it is necessary to be able to visualize it in 3-D space. While visualizing this data, engineers sometimes find it necessary to visualize changes in the characteristics of the flow with respect to changes in the value of certain input parameters. However, generating CFD data sets for several values of input parameters is a tedious procedure, and cannot be done interactively. This thesis proposes a method where approximations between data sets are used instead of resolving the entire CFD model so that interaction with the data can be performed interactively. There are three parts to this thesis. The first part describes the methodology and features of visualizing the data in a virtual environment through the interactive creation and manipulation of flow entities such as streamlines, cutting planes, and isosurfaces. The second part of the thesis analyzes two methods of approximation on large data sets. The third part, describes the actual interactive implementation of the approximation methods in a full scale application. The methods used here are the Virtual knot technique of B-spline curve fitting and basic linear interpolation. The input parameters, that are being varied, are the geometrical attributes of the data set. The application was developed using OpenGL and the Visualization Toolkit (VTK) libraries for the rendering and visualization calculations. In addition, the C2 interface libraries were used to create the Virtual Reality environment.

1. GENERAL INTRODUCTION

Introduction

Many diverse areas of scientific and commercial interest, involve the science of fluid dynamics. Engineers use this science to analyze and design aircraft, submarines, and industrial machinery. Scientists use it to investigate different natural phenomena such as weather and ocean currents and in building new devices. There are two methods of studying and analyzing flow, experimentally and computationally. Computational Fluid Dynamics (CFD) involves modeling a geometry and performing a numerical analysis on the flow through it, based on some input conditions. The process is usually complex and takes a large amount of time to generate a data set that contains information about the scalar and vector properties of the flow at a large number of discrete points. By this method, one may be able to predict the characteristics of a flow, without much experimentation (Munson et al., 1994). However, in order for this analysis to be beneficial, it is necessary to be able to visualize the results of the analysis.

There have been several software packages that perform 3D visualization of CFD data such as FAST (Bancroft et al., 1990). These results can be visualized on a computer desktop monitor, but do not provide a proper visualization as the desktop screen is two dimensional. Virtual Reality, however, offers a fully immersive and interactive environment, in which the user can physically move through the geometry while visualizing the data, thereby getting a much better knowledge and understanding of the flow. The first VR interface for visualizing CFD data were designed by Steve Bryson and Creon Levit in 1992 for examining air flow characteristics around an aircraft. Another such VR interface was built to FAST (Mahoney et al., 1995).

The generation of CFD data usually takes a large amount of time, of the order of hours. Many times, it will be necessary to analyze the flow at several different values of a single input parameter. In this case, this procedure becomes especially tedious, particularly when the results need to be approximate, and not extremely accurate. If we obtain data sets for certain

distinct values of an input parameter, we can approximate between the data sets and obtain intermediate data sets for any value of the input parameter. This will drastically cut down on the generation time, and numerical analysis procedures. The results might not be as accurate as an original result, but will still give the user an approximate idea of what to expect. If this methodology could be explored in a virtual environment, it would give the added advantage of being able to interactively visualize and compare two data sets. Approximation methods that could be utilized are B-spline curve approximation (Piegl and Tiller, 1997). Further improvisations to improve the accuracy could be made by utilizing fairing techniques as described in Hsieh and Chang, 1994.

This thesis describes methodology and features of interactive visualizing CFD results in a virtual environment through creation and manipulation of flow entities. It also analyzes the methods of approximation mentioned above for regenerating the data sets at intermediate parameter values. It goes on further to implement these techniques for interactive approximation and investigation in a virtual environment.

Thesis Organization

This thesis is organized into three papers. Each paper has an abstract, an introduction, and a body, followed by conclusions, results, suggestions for future work, and a reference list. A general conclusion at the end sums up the results of the three papers and the recommended suggestions for future work. A brief description of the organization of each of the three papers is described below.

Paper 1 (Chapter 2) : "Visualization of post processed Computational Fluid Dynamics data in a virtual environment". This paper gives a detailed description of the application for the virtual investigation of the CFD data. It describes the motivation behind the development, and follows up with a description of the hardware and software involved in developing the application. This is followed by a description of the application features such as interaction and visualization methodology. Here, all the flow entities that can be used for investigation of the flow, are described in detail. The paper also describes some of the problems encountered while dealing with multi-block structured data, and means of overcoming them. Finally, the

results and conclusions are elaborated with emphasis on user feedback, followed by several suggestions for future work.

Paper 2 (Chapter 3) : "Approximation of Computational Fluid Dynamics data for use in a virtual environment". This paper deals with the analysis of the approximation techniques, while performing them on large data sets with changes in input parameters. It begins with describing any previous work in the area, and follows with a detailed description of the approximation methods (Linear and B-spline curve fitting and fairing). Also described are the steps involved in pre-processing the data, followed by a numerical error analysis of the approximation methods. The results of the analysis are tabulated and this is followed by conclusions, and suggestions for future work.

Paper 3 (Chapter 4) : "Interactive Approximation of Computational Fluid Dynamics data in a virtual environment". This paper deals with the interactive implementation of the techniques that were analyzed in paper 2. This is done in an immersive virtual environment, where the input parameter can be interactively manipulated, and the data set regenerated spontaneously, by approximating between distinct values of the input parameter. An overview of paper 2 is given followed by a description of the hardware and software involved in implementing the approximation techniques. The input file configurations from the pre-processing are described, followed by a detailed description of the features used to investigate the approximation methodology, and compare it with distinct real time results, to check the accuracy of the approximations. Parallel programming is described in brief, to outline methods of improving the interaction in the virtual environment. This is followed by summarizing the results and suggestions for future work (Chapter 5) .

References

Bancroft, G. V., Meritt, F. J., Plessell, T. C., Kelaita, P. G., McCabe, R. K. and Glorus, A. "FAST: A multiprocessed environment for visualization of computational fluid dynamics," Visualization '90 Proceedings, IEEE Computer Society Press, Los Alamitos, CA, October 23-26, 1990, p.14-27.

Bryson, S. and Levit, C., "The Virtual Wind Tunnel: An environment for the exploration of three dimensional unsteady flows", IEEE Computer Graphics and Applications, v. 12, July 92, pp. 25-34

Hsieh, H. and Chang,W. "Virtual knot technique for curve fitting of rapidly varying data," Computer Aided Geometric Design vol. 11 1994 pp 71-95.

Mahoney, D. P., "Driving VR", Computer Graphics World, vol. 18, no. 5, May 1995, p. 22-35.

Munson, B. R., Young, D. F., and Okiishi, T. F, Fundamentals of Fluid Mechanics, 2nd ed., John Wiley and Sons, Inc., New York, N.Y., 1994.

Piegl, L. and Tiller, W. "The NURBS Book," 2nd edition, Springer Verlag 1997, N.Y.

2. VISUALIZATION OF POST PROCESSED CFD DATA IN A VIRTUAL ENVIRONMENT

A paper accepted by ASME Design Engineering Technical Conference

Vishant J. Shahnawaz, Judy M. Vance and Sasikumar V. Kutti

Abstract

This paper discusses the development of a virtual reality (VR) interface for the visualization of Computational Fluid Dynamics (CFD) data. The application, VR-CFD, provides an immersive and interactive graphical environment in which users can examine the analysis results from a CFD analysis of a flow field in three-dimensional space. It has been tested and implemented with virtual reality devices such as the C2, head mounted display (HMD) and desktop VR. The application is designed to read PLOT3D structured grid data and to display the flow field parameters using features such as streamlines, cutting planes, iso-surfaces, rakes, vector fields and scalar fields. Visualization Toolkit (VTK), a data visualization library, is used along with OpenGL and the C2 VR interface libraries, to develop the application. Analysts and designers have used VR-CFD to visualize and understand complex three-dimensional fluid flow phenomena. The combination of three-dimensional interaction capability and the C2 virtual reality environment has been shown to facilitate collaborative discussions between analysts and engineers concerning the appropriateness of the CFD model and the characteristics of the fluid flow.

Introduction

In various diverse areas such as aeronautical, chemical, civil and mechanical engineering, meteorology, naval architecture, oceanography and others, the analysis of fluid flow is central to research, design, and testing of facilities and products. A typical example would be the study of fluid flow around an airplane, which would be vital in determining its structural stability in flight. An understanding of the fluid flow around automobiles, aircraft and under-

water vehicles can be used to alter designs to reduce drag which in turn reduces fuel consumption. Engineers study the flow inside machinery like turbines, compressors and propellers to develop and improve the design of these machines. In each of these application areas, analysts are interested in examining scalar parameters of the flow such as temperature, pressure and density, as well as vector parameters, such as velocity.

Fluid dynamic analysis can be performed using actual parameter values obtained through experimentation or by theoretical computation. Computational fluid dynamics (CFD) is a technique for calculating approximations to the fluid flow and is commonly used in industry today to predict flow behavior (Munson, et al., 1994). The CFD results are generally displayed using desktop monitor systems. Existing CFD software tools provide limited VR capability. Ensight and Fieldview provide for stereo viewing, but not for three-dimensional input devices or multiple window synchronized displays.

In 1992, Steve Bryson and Creon Levit developed the first virtual reality (VR) application designed to allow users to examine fluid flow characteristics using immersive visualization and three-dimensional user interaction (Bryson and Levit, 1992). This application is called the Virtual Windtunnel and is used to examine the air flow characteristics surrounding various aircraft. Bryson continues to develop the capabilities of that application and intends to make a public release available in the near future (Bryson, et al., 1997). In 1995 Sterling Software Inc. and the Ford Motor Company developed a VR interface to FAST (Bancroft et al., 1990) (Mahoney, 1995). This capability was used by Ford engineers to examine the air flow around the engine of a vehicle.

In 1997, Brad Kohlmeyer, working under the direction of Jim Oliver at Iowa State University (Oliver et al.,1997), developed a VR application to examine CFD data which modeled the air flow from the air conditioning unit in the interior of a tractor cab. They used the C2 device at Iowa State University as the virtual environment. Kohlmeyer's application was developed as a demonstration of the use of VR for CFD and was not intended to be a general purpose VR CFD program. This paper outlines the capabilities of the VR CFD program developed at Iowa State University and presents initial user reaction of engineers and analysts who have used the application.

Motivation

Bryson and Levit's "Virtual Wind Tunnel" was developed to provide an interface where the results of CFD analysis on unsteady three-dimensional flow could be examined. Each time step represented a new flow pattern. These flow fields were pre-computed and a virtual reality interface was used to display these results. The hardware consisted of a BOOM^{TM} visual display, VPL data glove II with a Polhemus 3Space tracker, and a Silicon Graphics Iris 380 VGX computer. The Sterling Software/Ford application was implemented using the BOOM^{TM} visual display, Cyber Glove with a Flock of Birds tracker, and Silicon Graphics computers. One motivating factor concerning the work presented here was the desire to explore other virtual reality interface devices and environments for CFD analysis in order to facilitate collaborative design and discussion. The application discussed in this paper was developed for display in a head mounted display, on a stereo capable monitor and also in the C2 virtual reality room.

The C2 at Iowa State University is a room where stereo images are projected on three walls and the floor. Position tracking with the Flock of Birds is used to track one person's head and hand position allowing one person to act as a guide for the other participants investigation of the data field. Up to 12 people can be in the room comfortably at the same time. Each person wears CrystalEyes stereo glasses in order to see the stereo images being projected on the screens. Sound capabilities enhance the immersive nature of this device. The C2 is powered by two Silicon Graphics Onyx racks, each with 2 Infinite Reality Graphics pipes and 12-R10,000 processors. This device is most similar to the CAVE which was originally developed by Carolina Cruz-Neira (Cruz et al., 1993). Although the correct viewing perspective is only projected for the one person wearing the tracked stereo glasses, others in the environment rarely notice the small distortions viewed from their individual perspectives. Interaction within the C2 is provided by a wand or the Pinch Gloves. In the application developed here, the wand is used as the primary input device.

One key distinguishing characteristic of different virtual reality environments is the ability to immerse the user in the computer generated images. When users feel immersed in a vir-

tual environment they believe that the computer generated objects occupy specific locations in the virtual environment. This is accomplished through the use of head tracking and three-dimensional interaction. Field-of-view has also been shown to affect a person's sense of immersion in the virtual environment.

While the BOOM^{TM} provides excellent tracking and presents a non-intrusive head tracked visual display, it remains a one person device. In a collaborative environment, with an external monitor attached, participants can only describe and discuss what the user of the BOOM^{TM} cares to investigate. In addition, when using the BOOM^{TM}, only the person viewing the data through the BOOM^{TM} achieves any sense of immersion in the data. The head mounted display is similar to the BOOM^{TM} in that it also is a single person device. Only through the use of an external monitor can others participate in collaborative discussions of the data and the only person experiencing immersion in the data is the person wearing the helmet. The stereo capable monitor allows all of the participants to view and discuss the image without the aid of an external monitor, however, the sense of immersion that is felt when viewing stereo data captured on a desktop monitor is very low. The C2 provides an environment where multiple users are immersed in the data field and can easily discuss and investigate the entire flow field. The wide field-of-view and user controlled viewing allows multiple users to individually look around at the data and examine individual areas of interest.

The goal of this research was to provide a virtual environment for post processing CFD data which would allow stereo viewing and three-dimensional interaction in a collaborative environment. VR-CFD, a program that can be implemented using a variety of VR hardware, was built to meet these objectives. The following sections outline the structure and capabilities of VR-CFD.

Software and hardware

VR-CFD was developed using OpenGL, Visualization Toolkit (VTK), and the C2 library. The C2 library provides the virtual reality framework by providing functions which synchronize the four viewing walls to display one environment, display the correct viewing

perspective based on the head tracked position of the one pair of tracked glasses and allow interaction with the computer images through an input device. This library was developed by Carolina Cruz-Neira. OpenGL and the Visualization Toolkit are used to provide the graphics framework for this application. The VTK library provides functions for the calculation of fluid flow parameters and renders the output as OpenGL graphics (Shroeder et al., 1998). This software is download-able from the web at http://www.kitware.com/vtk.html. In cases where geometry display is needed and few calculations are required, OpenGL alone is used as the graphics software framework which provides faster rendering than the VTK software. Implementing VTK into the C2 library was fairly straightforward since both software libraries render graphical output using OpenGL. To get VTK output to display in the C2 environment, the VTK window widget is replaced with the X11 window widget of the C2 library. VR-CFD can be displayed using either the C2, the HMD, or a standard monitor with stereo display. Head and hand tracking exist in all these environments. Figure 1 shows a person interacting with the data in the C2 facility.

Figure 1. User interacting with data in the C2

Software Features

Interaction

Wand: This device is a joystick with a six degree of freedom position tracker attached such that the position and orientation of the wand can be determined at all times. The three buttons on the wand are configured to enable the user to perform functions such as translation, rotation, and scaling of the entire scene, and toggling between different modes of interaction for manipulating flow entities. The buttons are also used for interaction with a menu system which provides for selection of various tasks.

Menu System: The menu is an array of texture mapped 2D polygons that are attached to the front wall of the C2. Red buttons are used to open menus, close menus and switch between menus. Blue buttons either lead to further menus or activate functions. The menu organization was patterned after the Fieldview post-processing software so that users of Fieldview could easily navigate within the menu. Fieldview is a commercial post-processing software package. Interacting with the menu is accomplished by pointing the ray that extends from the wand at a button on the menu. Once the intersection of the ray with the button has been detected, the button is highlighted and can be selected using the trigger of the wand. See Figure 2 for a picture of a sample menu and wand cursor. When a button is selected, a sound is produced indicating positive selection

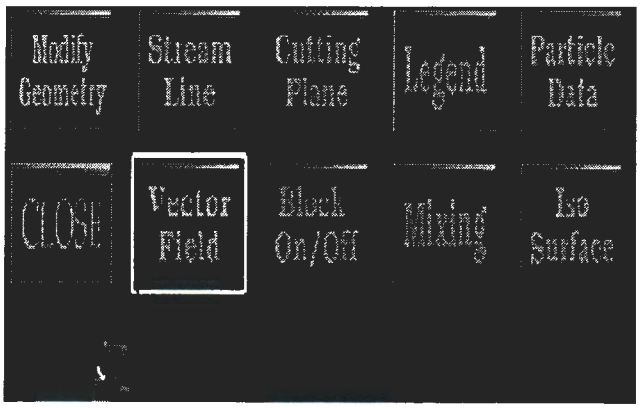

Figure 2. Menu system

Data format

At present the application supports two types of data, PLOT3D and Fieldview particle tracer format. PLOT3D is a standard data type for CFD applications.

Entity visualization

Streamlines: Streamlines in the flow indicate the path that a particle in the flow would take if placed in the flow at the seed point. The users have the ability to place seeds in the flow and create streamlines by moving into the flow and using the wand to indicate the seed position. Integration of the velocity vector field can be performed forward, backward or in both directions through the flow-field from the seed point. Figure 3 shows two rakes of streamlines. Rakes are a set of streamlines originating from a source line. The streamlines can be displayed as either lines, tubes or ribbons. Scalar parameters such as pressure or temperature can be color mapped onto the streamlines. Multiple streamlines and multiple rakes can also be placed in the flow

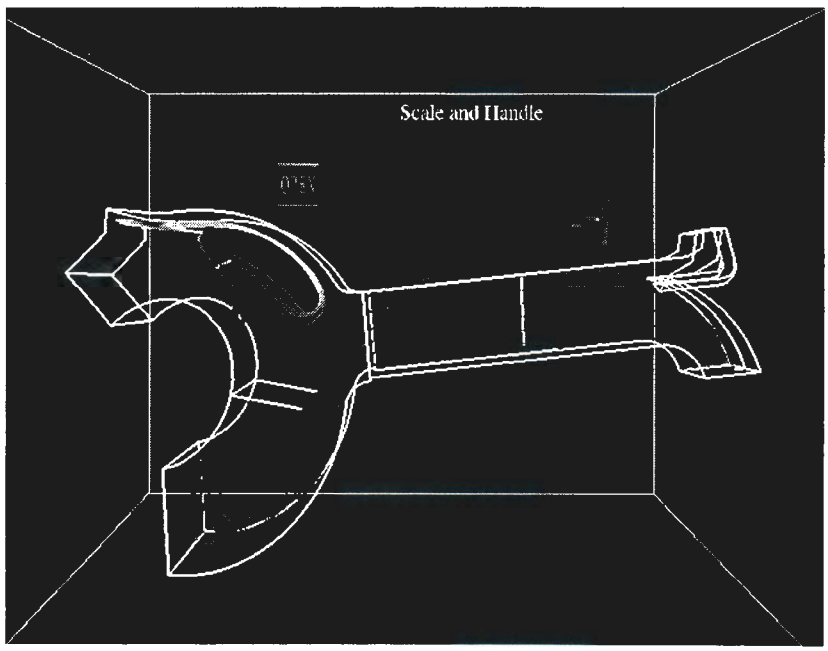

Figure 3. Rakes of streamlines

Animated flow particles: Bubbles placed on the streamlines can be animated to better show the path of particles in the flow (see Figure 4). The streamlines are set as the paths for tracer particles. If multiple streamlines are selected as paths, bubbles start out on each streamline at the same time. These bubbles, however, travel at the instantaneous speed of the streamline as they move along the streamline. Examining the movement of bubbles on multiple streamlines gives the user an indication of the relative velocities of areas of the flow. The bubbles can be animated both forward and backward.

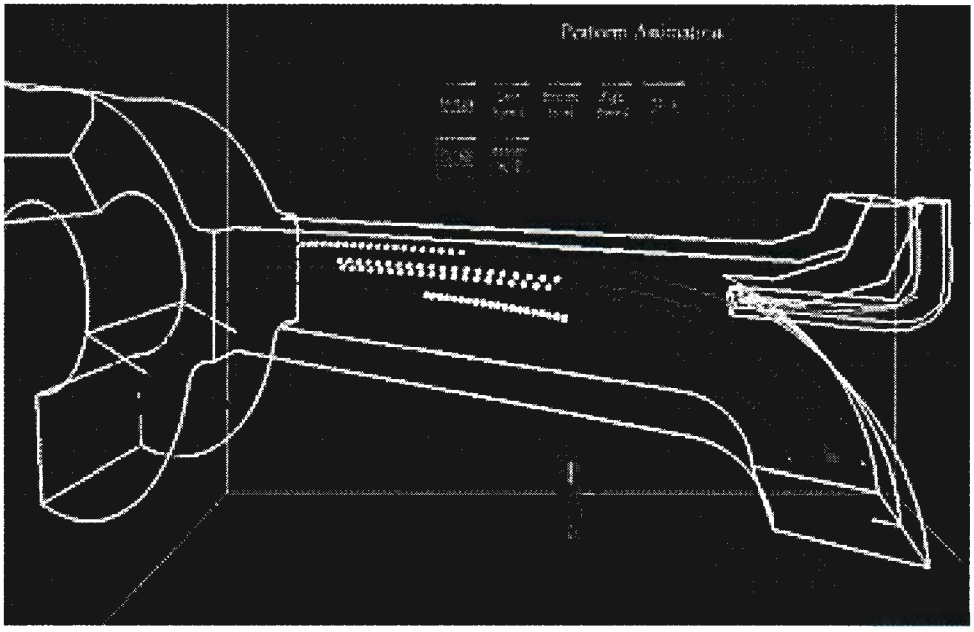

Figure 4. Animated bubbles along streamlines

Cutting Planes: Cutting planes can be oriented according to the local x, y, and z axes or interactively placed at any orientation. The planes can be color mapped according to the chosen scalar value. The default color mapping selects a range based on the maximum and minimum values of the scalar. Often, the area of interest is contained in a narrow range of scalar values. In order to more clearly visualize different areas of interest, the user can adjust the color mapping maximum and minimum values. All areas below and above the range are set to the maximum and minimum values of the range respectively. The range of values can be inter-

actively defined using a slider mechanism. In this way, the user can more clearly differentiate the data in the range of values of interest(see Figure 5) . In-plane vectors (see Figure 6) can also be mapped to the cutting plane.

Vector Field: Capability has been provided for visualizing the full vector field. Lines are drawn at every point and are oriented along the direction of the vector at that point. The field is color mapped according to the vector magnitude. (see Figure 7).

IsoSurfaces: Isosurfaces are surface contours of points having the same scalar value. Users can visualize isosurface pairs at a minimum and a maximum value. The pair values can be changed and defined by the user. The user can also switch between scalars, with the maximum and minimum range values as the defaults. (see Figure 8).

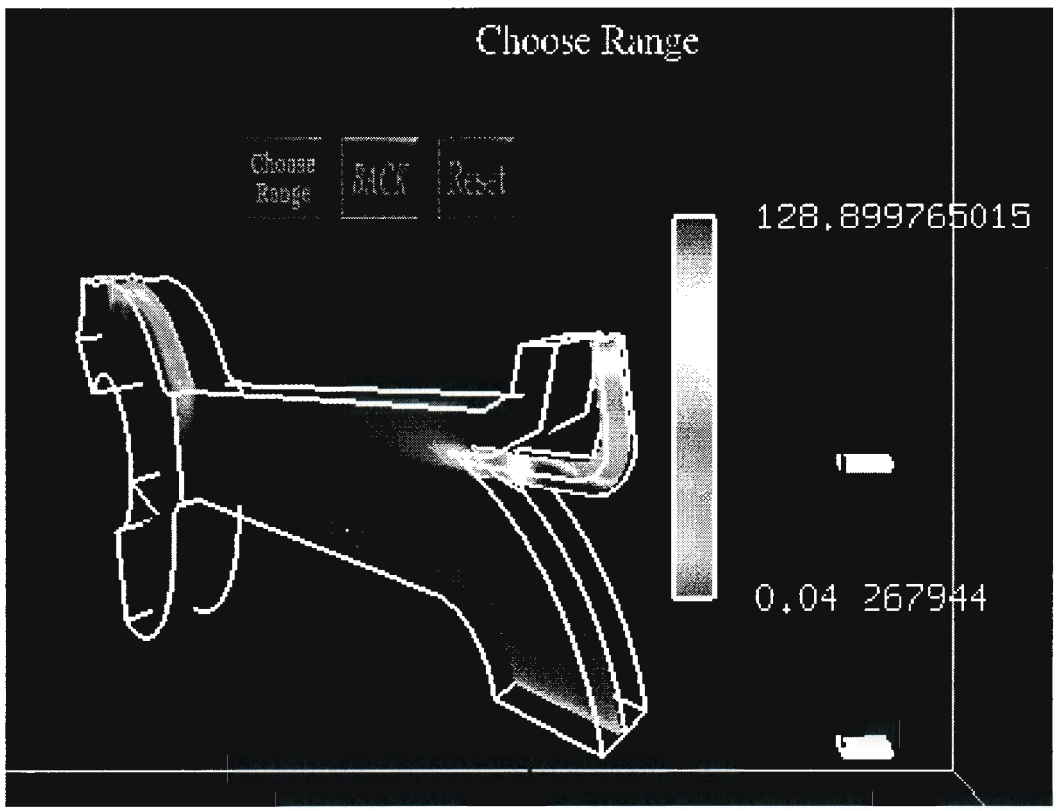

Figure 5. User-defined color map ranges

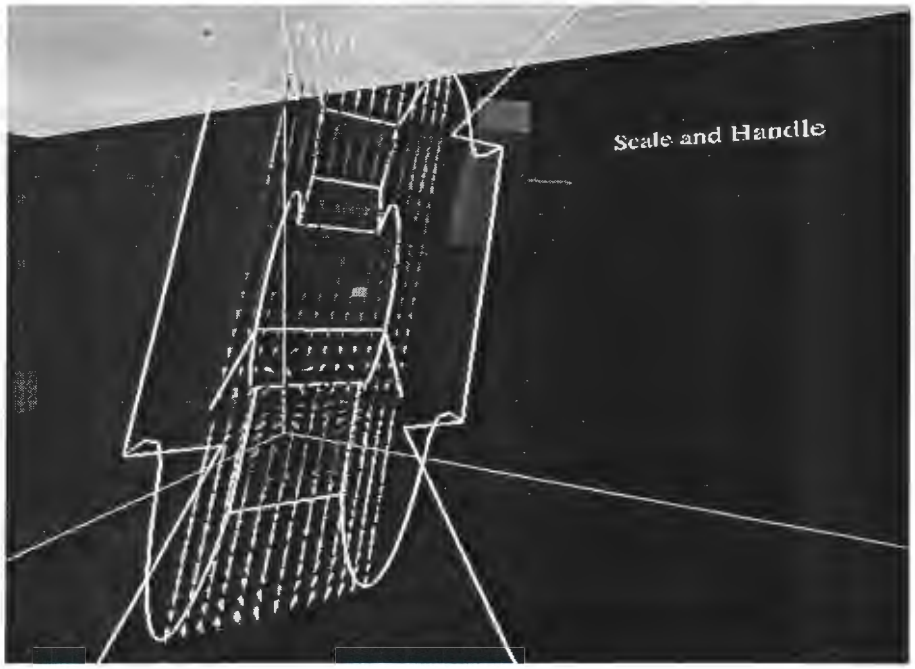

Figure 6. In-plane vectors

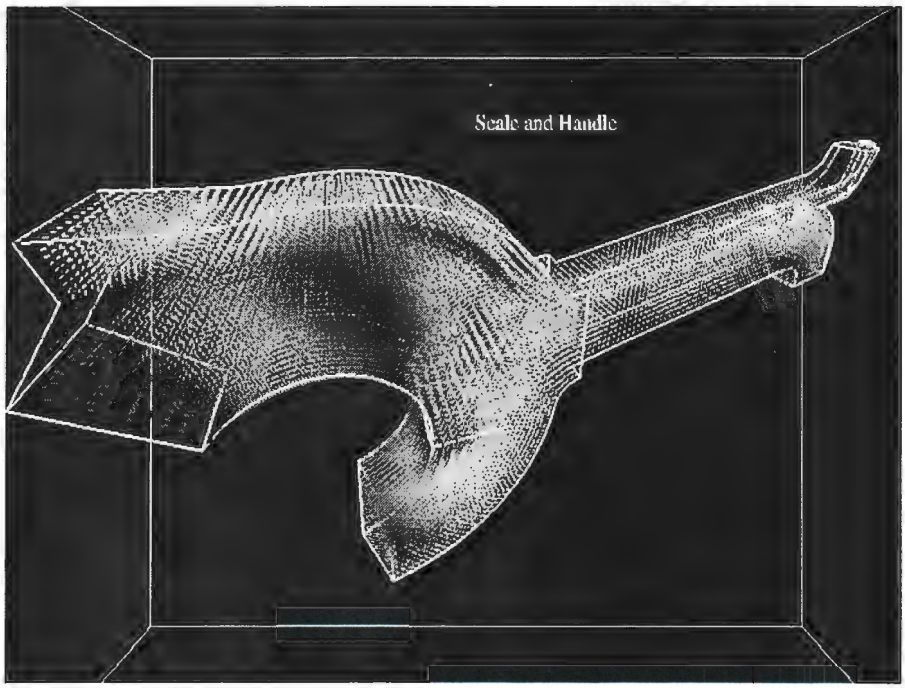

Figure 7. Full field velocity vectors

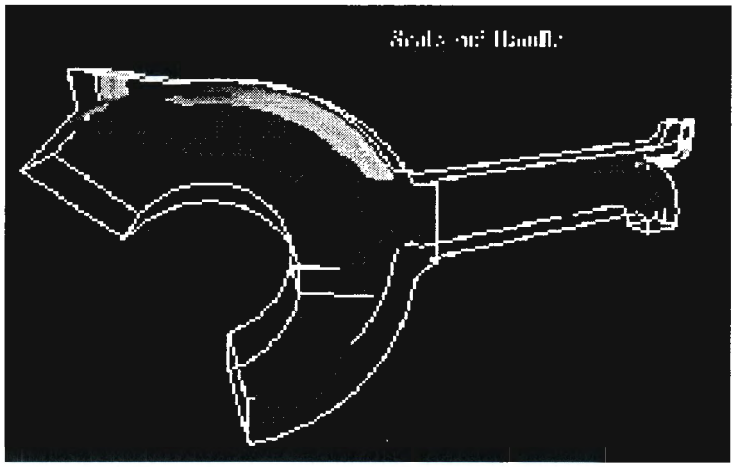

Figure 8. Two isosurfaces of scalar parameters

Mixing: In flows that involve mixing of two or more streams, it is helpful to visualize the amount of mixing between two streams. In Figure 9, the first fluid is shown as green and the second fluid is red. As the two streams mix, they form areas of dominant red, green, or yellow as well as tints of reddish and greenish yellow depending on the degree of mixing. This color identification enables the users to locate areas of interest where the mixing is good, unequal etc. For example, in Figure 9 it can be observed that there is an almost equal mixing of the red and green streams in the upper central region of the geometry.

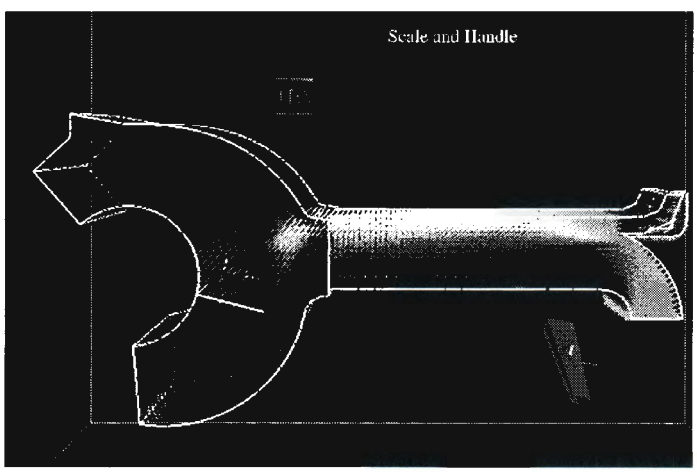

Figure 9. Mixing of two or more streams

Multiblock data

It is usually not possible to generate a single structured grid for a complex flow field. Different areas of the flow require finer meshing. Unstructured grids can be used in these cases, but the computation time for creating flow entities while using unstructured grids is quite high. To overcome this difficulty, it is common to divide the flow field into several smaller structured grids of varying sizes. Each division is called a block and the flow field data is stored as multiblock data. This allows flexibility in modeling while providing for faster computations.

The VTK software does not specifically support multiblock data format. For entities such as cutting planes and isosurfaces, a given entity most likely will span several blocks of data and can be calculated in each block independently. However streamline generation presented the problem of creating continuous streamlines extending from one block to another. The data that was supplied to us was multiblock data. One approach to overcoming this limitation with generating streamlines in multiblock data using VTK was to append together all of the structured grid blocks and store the result as one unstructured grid data file. Theoretically it should be possible to generate all the visualization components with the resulting unstructured grid. But VTK software has an error which prevents the formation of streamlines when there are coincidental points making up flat hexahedral cells, which result at the common surface of two blocks of data when they are appended. This approach to dealing with Multiblock data was abandoned.

Results

Over the course of the development of this software several meetings were held with the sponsors of the research. Typically these meetings involved five people: two of the co-authors of this paper, a manufacturing process engineer, a CFD analyst, and a CFD code developer. User response to the VR CFD application paralleled results reported by Bryson, et al. (1995) which states that users were enthusiastic about the capabilities that VR provided for examining fluid dynamics data. Bryson reports that users who viewed known data sets found flow

phenomena which hadn't been recognized before. It was our experience that in addition to this, users were able to more fully reconfirm their understanding of the fluid flow while working in the virtual environment. The C2 virtual environment provided a fully immersive environment in which to examine the results of the data analysis. Users were comfortable moving in and out of the flow to examine the flow characteristics. It was common for the team to spend up to 3 hours at a time in the C2 and up to 6 hours per day examining the data. Interaction with the wand was very natural and easy to learn. The menu hierarchy was familiar because it was organized similar to the Fieldview menu organization.

During one visit, the application was demonstrated on both the stereo monitor and in a head mounted display. The user interaction with the wand was the same in all three environments and head tracking was also implement in all three environments. Feedback from the users indicated that the head mounted display was intrusive and did not encourage collaboration in examining the data flow. The stereo monitor's lack of ability to provide the user with a sense of immersion greatly reduced the advantages of implementing this application using virtual reality. The use of the C2 encouraged collaboration and was a comfortable interface in which users could work for an extended period of time.

Future Work

There are many ways to improve this application. The following avenues are currently being explored:

Generality: The generality of this application as a post-processing tool can be further improved to be able to specify a larger number of data formats, and to take in a larger number of scalars and vectors.

Computation speed: This is a key issue in CFD post-processing. CFD grids can easily become very large and trade-offs must be made between displaying all of the data and maintaining real time display speed for VR on the order of 12-15 frames per second as a minimum. The application presented here contained 50,000 cells. When the full velocity field was displayed, the frame rate dropped below 10 frames per second. Displaying multiple rakes with a

cutting plane also reduced the frame rate. Researchers at the National Center for Supercomputing Applications are developing an interface where VTK and Iris Performer are combined in an effort to increase display performance (Leigh et al., 1998). Bryson and Gerald-Yamasaki (1992) have investigated implementation of a distributed architecture for the Virtual Windtunnel. Both of these approaches will be investigated in the near future.

Calculation of new scalar and vector properties: This capability will allow users the ability to define more parameters by using the given scalar and vector data.

Flow field approximation: Since CFD simulations take a long time to generate, approximating, interpolating or extrapolating the flow fields using intermediate existing flow fields would be an effective way of obtaining an understanding of the flow behavior with changes in parameters such as geometry and time. This approach has been successfully applied to finite element results (Yeh and Vance, 1998) and similar methods will be pursued as they relate to CFD data.

Multiple usage and VR networking: This development will enable different users in different locations to interact with the application at the same time. At present, the application can only run in one location. In the future, several users at different locations in different parts of the world will be able to work with the same data.

Menu system: Currently, the menus are stationary. There are many different menu paradigms possible in VR. Different options such as the VUI developed by Daniel Heath (1998) will be investigated.

Conclusions

A virtual interface tool (VR-CFD) for the visualization of CFD data was developed. Features for creation and manipulation of flow visualization entities, and for interacting with the geometry were provided. These entities include streamlines, rakes, cutting planes, isosurfaces, and vector fields. Interaction was provided using the wand and through a menu hierarchy. Other features included visualization of full field vectors, mixing streams and animating tracer particles. Data formats supported include PLOT3D and Fieldview particle data. The flow visu-

alization functions of VTK were used for calculations and rendering along with OpenGL. The VTK library was integrated with the C2 library which was used as the virtual reality platform.

The C2 virtual reality environment provided a comfortable three-dimensional immersive environment for the examination of the results of CFD analysis. This virtual environment facilitated collaboration between various members of the design/analysis engineering team.

Acknowledgments

The authors would like to acknowledge the support of Procter and Gamble Inc., Cincinnati, Ohio and the Iowa Center for Emerging Manufacturing Technology, Iowa State University, Ames, IA.

References

Bancroft, G. V., Meritt, F. J., Plessell, T.C., Kelaita, P. G., McCabe, R. K. and Glorus, A. "FAST: A multiprocessed environment for visualization of computational fluid dynamics," *Visualization '90 Proceedings*, p.14-27, IEEE Computer Society Press, Los Alamitos, CA, October 23-26, 1990.

Bryson, S., Johan, S., and Schlecht, L., "An Extensible Interactive Framework for the Virtual Windtunnel," *1997 Virtual Reality Annual International Symposium Proceedings*, Albuquerque, NM.

Bryson, S., Johan, S., Globus, A., Meyer, T., and McEwen, C., "Initial User Reaction to the Virtual Windtunnel," *American Institute of Aeronautics and Astronautics Meeting Proceedings*, Jan 1995, Reno, NV.

Bryson, S. and Levit, C., "The Virtual Wind Tunnel: An environment for the exploration of three dimensional unsteady flows", *IEEE Computer Graphics and Applications*, v. 12, July 92, pp. 25-34.

Cruz-Neira, C., Sandin, D. J., and Defanti, T. A,. "Surround screen projection based virtual reality: The design and implementations of the CAVE", *ACM SIGGRAPH 1993 Proceedings*, vol. 27, August 1993, p.135-142.

Heath, D. J., "Virtual User Interface (VUI) A windowing system for VR", *Second International Immersive Projection Technology Workshop Proceedings*, Iowa State University, Ames, IA, May 11-12, 1998.

Leigh, J., Rajlich, P. J., Stein, R. J., Johnson, A. E., and DeFanti, T. A., "LIMBO/VTK: A tool for rapid tele-immersive visualization," *IEEE Visualization '98 Proceedings*, October 18-23, 1998, Research Triangle Park, NC.

Mahoney, D. P., "Driving VR", *Computer Graphics World*, vol. 18, no. 5, May 1995, p. 22-35.

Munson, B. R., Young, D. F., and Okiishi, T. F, *Fundamentals of Fluid Mechanics*, 2nd ed., John Wiley and Sons, Inc., New York, N.Y., 1994.

Oliver, J., Vance, J., Luecke, G., and Cruz-Neira, C., "Virtual Prototyping for Concurrent Engineering", *International Immersive Projection Technology Workshop Proceedings,* Stuttgart, Germany, July 5, 1997, p. 51-57.

Schroeder, W., Martin, K. and Lorensen, B., *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Prentice-Hall, Inc., New York 1998.

Yeh, T.-P., and Vance, J. M., "Applying Virtual Reality Techniques to Sensitivity-based Structural Shape Design," *Journal of Mechanical Design*, vol. 120, December 1998, pp. 612 - 619.

3. APPROXIMATION TECHNIQUES FOR COMPUTATIONAL FLUID DYNAMICS DATA FOR USE IN VIRTUAL REALITY

A paper to be submitted to the 2000 ASME Design Automation Conference.

Vishant J. Shahnawaz and Judy M. Vance

Abstract

Virtual Reality (VR) provides a fully interactive three dimensional interface in which users can interact with computer generated models. One application recently developed is a virtual reality environment for examining the results of computational fluid dynamics analysis (CFD). In an effort to extend the functionality of that application beyond its use as a post processing tool to that of a design tool, CFD approximation techniques have been investigated. This paper presents two approximation techniques that can be used to approximate the flow characteristics of a fluid as changes are made to the model geometry. The virtual knot technique of B-spline curve fitting is presented here and applied to two example problems. Numerical analysis is performed to compare the accuracy of the approximations to the actual solutions. The results indicate that either approach would be satisfactory for implementation in a virtual environment, however, the accuracy of the approximations is closely related to the number and spacing of the actual solutions which are available and the complexity of the flow field.

Introduction

Computational fluid dynamics analysis (CFD) is a numerical tool commonly used to calculate the flow parameters of a fluid. The fluid properties, initial conditions, boundary conditions, and the shape of the geometry surrounding the fluid affect these flow parameters. With ever increasing computational power available, analysts are modeling increasingly complex three-dimensional flow fields. The solutions to these models often take hours or days to compute, even on large multi-processor computers. Because of this, fluid flow design is not an iter-

ative process. Designers are hesitant to explore new ideas because of the cost, both in time-to-market and computer resources that are required to investigate a large number of designs.

Recently a virtual environment was developed to allow analysts and designers to interactively examine the three dimensional model geometry and CFD analysis results. This environment provides three dimensional interaction tools and stereo viewing which enables users to interact more naturally with three dimensional data. Cutting planes, iso-surfaces, streamlines and rakes can be placed in the geometry and the resultant display will reflect the results of the CFD analysis. A full description of this application can be found in the reference by Shahnawaz and Vance (1999).

The authors have observed users in the VR environment posing many "what-if" questions related to how changes in the geometry of the model would affect the fluid flow characteristics. Any change to the model geometry requires a new CFD solution, which takes a considerable amount of time to compute. This is the motivation for the work presented here which involves examining approximation techniques for CFD data which would be suitable for implementation in a virtual environment.

In this paper, two methods are applied to the example problems and analyzed for accuracy. The virtual knot technique (Hsieh and Chang, 1994) fits a cubic B-spline curve through a set of actual solutions that have been calculated at various design configurations within the possible design space. The linear interpolation technique uses a straight line interpolation between two pre-computed solutions. In the examples presented here, only one design variable, a change in the model geometry, is used.These approximation methods are easily extendable to other input parameters, such as initial conditions and boundary conditions.

Previous work

Approximating flow fields has so far, involved generating a denser vector field, from random sample points obtained from numerical simulation, or experimentally. This typically involved two dimensional vector interpolation, with the aim of preserving vector magnitude (Schaffer and Doswell,1978). While it is possible to linearly interpolate the individual components of the vector, and obtain a correct direction of the vector field, the resulting magnitude

will be in error. The method proposed, was to interpolate vector fields using divergence and vorticity as constraints, and thereby preserving them. Another method implemented was that of the "one-step" and "two-step" quadratic least squares fitting (Zhong et al., 1992). This method involves using constraints like divergence to interpolate a vector field. It was found to produce good results for small data sets of 500 points and is insensitive to small deviations or "noise". This method has been extended to preserve the isotropic turbulence property and isotropic homogeneous turbulence property (Zhong et al., 1993). Kuroe et al (1998) proposed a method for approximating the vector field using neural networks.

The common feature that can be seen in all of the above interpolation or approximation methods is that they were used to generate a single, complete vector field for a flow model (two-dimensional or three-dimensional), by interpolating or approximating between sample points that were either computer generated or measured at discrete points in that flow model. In this paper, however, we have complete vector fields, that have already been generated at distinct values of an input design variable of the flow model. We need to be able to approximate between these flow fields, in order to be able to obtain a new vector field for any value of the input design variable. The methods mentioned above could be extended to solve this problem, however, they could involve a lot of computation power, and the extension process might not be trivial. Hence, we focus on simpler, more direct methods, such as linear and spline interpolation, and as future work, could extend the methods described above to get a possibly more accurate solution.

Approximation methods

Terminology

There are various approximation and interpolation methods available. We begin by defining some of the terminology involved in this field. For further details see (Piegl and Tiller, 1997).

Interpolation: Fitting a curve or straight line through a set of points with the curve passing through all the points.

Approximation: Fitting a curve or straight line through a set of points without the curve passing through all the points. Instead the curve approximates the points passing close to them, minimizing the error by which it misses them.

Parametric form: Expressing a two-dimensional or three-dimensional vector **x** as functions of a single parameter, *u*.

$$\mathbf{x} = (x(u), y(u), z(u)) \ldots\ldots (1)$$

Implicit form: Expressing a two or three dimensional vector in the form of a function f(x,y,z) = 0.

B-spline curve: This is a piecewise polynomial curve, the shape of which is affected by entities known as control points.

Basis Functions: These are the functions of the common parameter u. The curve is expressed as a linear combination of these functions.

Control points: These are the points that affect the shape of the curve. The movement of a control point results in a change in the shape of only the parts of the curve affected by that control point.

Knots: These are break point parameters at which the control points are calculated. This splits the curve into separate pieces, which are manipulated by the associated control points.

Point Inversion: This is the process of calculating the parametric value, u, given the vector component function x(u).

Linear interpolation

The common linear interpolation method is summarized as follows.

If *s1* and *s2* are the function values at points *p1* and *p2*, then at any point *p*, such that *p* lies between *p1* and *p2*, the function value *s* can be calculated by the formula,

$$s = s1 + (s2 - s1)(p - p1)/(p2 - p1) \ldots\ldots (2)$$

B-Spline interpolation

In case of B-spline interpolation, we use the parametric form of interpolation. The B-spline curve is expressed as

$$C(u) = \sum_{i=0}^{n} f_i(u) P_i \quad \text{..... (3)}$$

Where n is the total number of control points, $\mathbf{P_i}$ are the control points, and $f_i(u)$ are the B-spline basis functions. The B-spline basis functions are expressed as

$$N_{i,0}(u) = \begin{pmatrix} 1, if\, u_i \le u < u_{i+1} \\ 0, otherwise \end{pmatrix} \quad \text{..... (4)}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \quad \text{..... (5)}$$

where the u's correspond to the knot vector $\mathbf{U} = \{u_1, u_2, u_3, \ldots u_n\}$ and p is the degree. For further details see (Piegl and Tiller, 1997).

Curve fairing or smoothing

When fitting a curve, it is observed that wherever there are large discrepancies or gaps in the data, the curve tends to "wiggle" or experience wild deviations. This can cause a lot of errors when checking against the real solutions at intermediate points. We seek to minimize this error and "wiggling" in order to fit a smooth curve between the points. This process is called curve fairing.

A lot of work has been done in the field of curve fairing. Kjellander(1983) proposed a method for interactive smoothing of parametric splines. This is done by enforcing curvature derivative continuity and third degree continuity. It involves moving the data points around to get a smoother curve. This method however does not work in cases where the points have to be preserved. Poliakoff (1995) extended this method to 3D non-uniformly parameterized curves. Lott and Pulin (1988) developed a similar method for surfaces using nonlinear optimization.

An algorithm for local fairing of B-spline curves was proposed by Sapidis and Farin (1990). This was done by enforcing certain "fairness criteria". The first criterion stated that "a curve is characterized as fair if its corresponding curvature plot is continuous, has the appropriate sign and is as close as possible to a piecewise monotone function with as few as possible monotone pieces". The second criterion stated was "A B-spline curve should always be faired at the offending knot location, i.e. at the knot location corresponding to the largest curvature discontinuity". This meant that the first step was to determine which knot produced the highest curvature discontinuity. This knot could then be removed and new control points could be calculated. This process also requires that the original data point be moved.

Nowacki and Lu (1992) proposed a method for fairing polynomial curves by applying constraints to approximation conditions, end conditions and an integral condition pertaining to the area under the curve while minimizing a linear combination of the square of the second and third derivatives. Hamman and Chen developed a method that involves identifying the right data points for curve approximation when presented with an large number of points through which to approximate.

The Virtual knot technique was developed by Hseih and Chang (1994) and is used in this work. Details of the method are presented next.

Virtual knot interpolation technique

The virtual knot technique involves calculating the spans between the data points, and then fitting in virtual data points, or "virtual knots" in the gaps between the data points. An arbitrary number of knots can be selected for insertion, and the algorithm calculates the places where the knots are to be inserted. The type of curve used here is a cubic B-spline curve. The drawing illustrates the procedure. The thin line is the unfaired curve. The thick line is the faired curve. The implemented technique is briefly outlined in this paper. For more details, see (Hsieh and Chang, 1994).

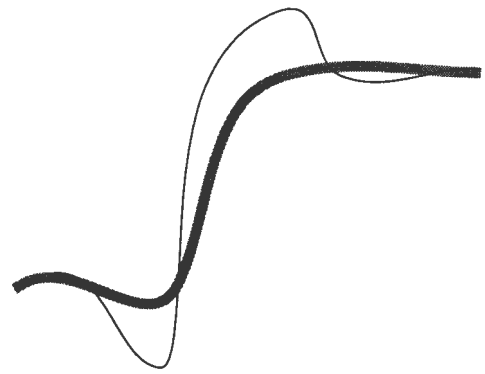

Figure 10. Faired and unfaired curve

Consider a set of data points

p_1 p_2 p_3 p_4 p_5 (gap in data) p_6 p_7 p_8

It is desired to calculate additional knots to be inserted into the gap in the data in order to get a better approximation of the curve. After the knots are inserted we get

p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12

The method takes as input, the total number of pre-specified data points N_r for the curve which should be greater than the actual number of data points. The geometric span can be defined as the length or distance between two points.

$$d_i = \sqrt{(p^2 xi+1 - p^2 xi) + (p^2 yi+1 - p^2} \ldots\ldots (6)$$

Let d_I be the geometric span between points p_i and p_{i+1}. Then the number of virtual data points is calculated to be

$$n_i = \left\lfloor \frac{d_i}{\sum d_i} N_r \right\rfloor - 1 \qquad \text{..... (7)}$$

Hence, we have a larger number of data points including the virtual data points. The distribution of the virtual data points depend on the geometric span. If the span is larger, a larger proportion of the virtual data points are inserted into the span. Then the control points associated with these data points can be calculated. This is done by solving the system of linear equations.

$$(T + \beta U^T M U)v = \beta U^T M \hat{x} \qquad \text{..... (8)}$$

where β is the weighting factor, which determines whether the curve is an approximation or interpolation. At a high value of β (greater than 10000) the curve passes through all the actual data points. T is called a *relation matrix* and U is called a *couple matrix*. These are derived by minimizing a function that consists of the energy functional of the curve and the error functional. The energy functional refers to the "bending energy" of the curve, and the error functional refers to the error between the actual data points and the points on the curve (See Hseih and Chang, 1994 for details). The vector **v** is the control point vector, and the vector **x** is the data point vector which contains the original data points and zeros, in place of the virtual data points. The matrix **M** is a mark up matrix, which has diagonal elements as 0, if the index corresponds to a virtual data point, and 1, if the index corresponds to an actual data point respectively. When the vector **x** is multiplied with the vector **M**, a final data point vector is obtained, and is used to solve for the control points.

Point inversion

In case of a least squares approximation, or any other polynomial approximation/interpolation method, for a given desired point, P, we obtain the coefficients of interpolation, and hence can obtain the function *f(x)* at any given value *x*. This method, however, only works for

implicit form equations. For a B-spline curve, which is in parametric form, two steps need to be performed.

a) For a desired point, the corresponding parametric point, u, is calculated.

b) The value of the function $f(u)$ is calculated.

The second step is easily performed once u is found. The first step, however, involves a process called point inversion (Piegl and Tiller, 1997). Let the point **P** = *(x,y)*, lie on the B-spline curve **C**(u). Point inversion involves finding the value u, on **C**(u) which lies closest to **P**,. This is an iterative process, involving minimizing that distance between **C**(u) and **P**. The iterative process used is Newton iteration. Let the start value be u_0. This is the value of u which is closest to **P**. **P** is denoted by $\mathbf{C}(u_r)$. The dot product function between **C**'(u) and **C**(u)-**C**(ur), is formed.

$$f(u) = C'(u) \bullet (C(u) - C(u_r))) \quad \text{..... (9)}$$

We need to minimize this function, and find the value of u. This will end up being closest to u_r. A defined number of iterations are performed, or a convergence requirement is prescribed. If u_i is the parameter obtained at the i-th iteration, then u_{i+1} can be calculated as follows.

$$u_{i+1} = u_i - \frac{f(u_i)}{f'(u_i)} = u_i - \frac{C(u_i) \bullet (C(u_i) - C(u_r))}{C''(u_i) \bullet (C(u_i) - C(u_r)) + |C'(u_i)|^2} \quad \text{..... (10)}$$

The convergence criteria is given by eqns. (11) and (12). ε is a certain minimum value.

$$|C(u_i) - C(u_r)| \leq \varepsilon \quad \text{..... (11)}$$

$$\frac{|C'(u_i) \bullet (C(u_i) - C(u_r))|}{|C(u_i)\| C(u_i) - C(u_r)|} \leq \varepsilon \quad \text{..... (12)}$$

Once the value of u_i satisfies the convergence criteria, it can be used to calculate the value of the function f(x(u)) to find the function value corresponding to x.

Data pre-processing

Data handling

The data that is used in this study is output from the CFD analysis code in PLOT3D structured grid format. A structured grid is composed of many cells. Each cell is uniform in shape and is defined by eight points contained in the grid. the total dimension of the grid is i x j x k. Two types of files are needed, a geometry file (PLOT3D grid file), and an output data file. The grid file has the following information. a) Number of grids(n) b) Local dimensions of each grid (i x j x k) c) (X,Y, Z) points of grid 1, (X,Y, Z) points of grid 2,... (X,Y, Z) points of grid n. The output data file contains data in a similar format with a) Number of grids b) Local dimensions of each grid (i x j x k) and the number of scalars c) Scalar data for the grid points of grid number 1...n.

One of the difficulties encountered when trying to approximate between solution sets is that re-meshing could have occurred. If the mesh remains the same in all solutions, interpolation of the scalar and vector values can take place between solutions at each grid point. If re-meshing has occurred, then a new approach must be followed. The approach followed here is to keep one case as a template dimension structure, and probing the corresponding grids in the other cases using this template. We further elaborate how this case is handled.

Let grid 1 of case 1 have a structure of 10 x 10 x 10 and grid 2 of case 2 have a structure of 12 x 12 x 12. Consider grid 1 of case 1. We know the bounds of the grid 1 of case 2. We hence, generate a new grid with bounds of case 2, but with the dimensions of case 1. This yields a new grid of case 1 dimensions in the same space as case 2. Next we use the linear interpolation libraries of the Visualization Toolkit (Schroeder, 1998) to determine the scalar and vector values at the new grid locations in case 2, thus forming a modified data set for case 2. This modified data is used as the case 2 solution set for the interpolation between case 1 and case 2.

Approximation/interpolation process

Two methods of approximation/interpolation are implemented in this work. In the case of linear interpolation, no preprocessing of the output data needs to be performed. However, in the case of the Virtual knot technique of B-spline curve fitting, (henceforth referred to as simply spline interpolation/approximation), the set of control points needs to be calculated for every point in the data set. This is performed as a pre-processing step and the number of control points and the control points themselves are stored in data files. Each scalar has its own control point data file. The control points obtained are of the form **p**(input parameter, scalar).When an arbitrary value of the input parameter is picked, the point inversion process uses the control points to calculate the parametric value on the curve, and then calculates the scalar value using that parametric value.

Numerical analysis

The analysis was performed on two different geometries. Pictures of each are shown in Figures 11-14. In Figure 11., fluid flows in from the top the long duct, and another fluid flows in from the top of the short duct and mixes at the point of intersection, and travels down the rest of the length of the longer duct. The angle between the short duct and the long duct is the parameter that is being changed. Hereafter, this will be referred to as the 'duct' data set. In Figures 12-14, the fluid flows in from the left end, over the step and passes out through the right end. Here, the length of the step over which the fluid flows is being changed. Hereafter this will be referred to as the 'step' data set. The figures show only a few sample cases through which the approximation is performed. The first case does not have much of vorticity. The second case sees a lot of vortical action in the area just in front of the step.

We would like to be able to interpolate between the cases, and generate new flow fields for any intermediate parameter. Finally, we would like to be able to visualize streamlines, cutting planes and other entities in a virtual environment as is done in our previous work on visualization (Shahnawaz and Vance, 1999), but using the new flow field. In this paper, we will focus on the numerical analysis part.

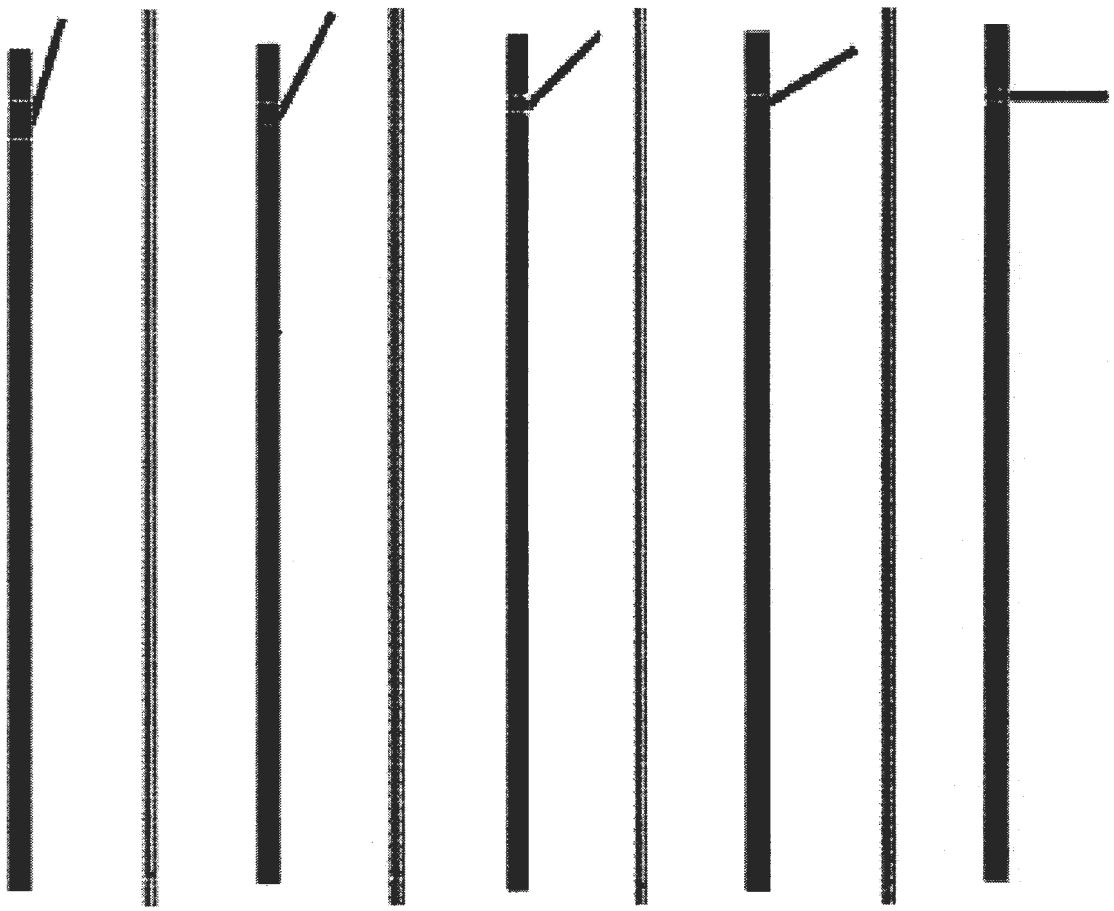

Figure 11. Duct with different angles

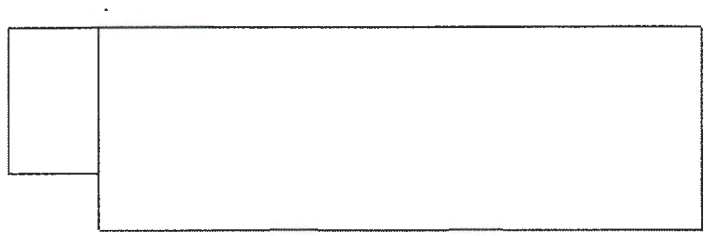

Figure 12. Step of length 3 m

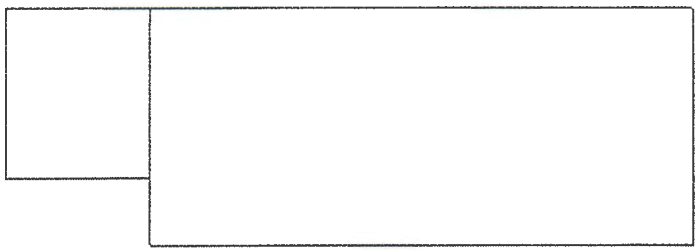

Figure 13. Step of length 6 m

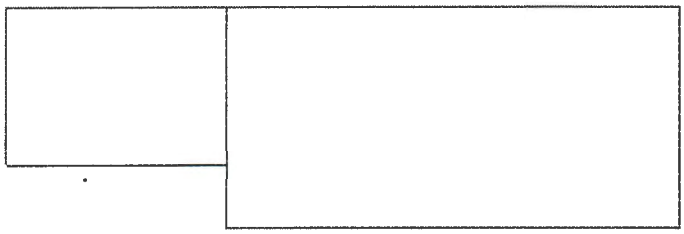

Figure 14. Step of length 8 m

Comparison methodology and criteria

The comparison consists of comparing an approximated flow field and an original flow field. There are two methods of comparison. One method is by visually comparing the data as displayed in the virtual environment. Another method is numerical comparison. In this paper, we focus only on numerical comparison.

For the comparison, the output data control point files are read into a data structure. Then, for a given known angle/step-length, the flow field is generated. The results of the approximation for each scalar or vector value are then compared to the analysis data set generated at that angle.

There are two comparison criteria that can be used. One, is the percentage error or deviation averaged over the entire data set. The other, is the number of noise points, or those points that have deviations greater than 25%. After looking at both of these criteria and both approximation methods, the user can select the most appropriate method for the given data set. Since

both linear interpolation and virtual knot insertion technique are available, an initial comparison can be performed to guide the user as to which method to use on a given data set.

Analysis results

Step data

In the case of the step data it was found that generally, the spline approximations worked better than the linear approximation. The approximation was done over step lengths of 3.0, 4.0, 4.75, 6.4, 7.25, 7.75 and 9.0. Original solutions were obtained through CFD analysis at step-lengths of 5.25,6.75 and 8.25. These were used for accuracy checking. The linear approximation compared favorably with the actual solutions. The total number of points in this data set is 15620. The results are summarized below in Table 1. Comparison criteria are given for the u, v, and w velocity components and the pressure Pr

TABLE 1. Step length approximation analysis

Step length & Scalar	Average % Error Linear method	Average % Error Splines method	Number of noise points Linear method	Number of noise points Splines method
5.25 -- u	1.1780	0.0550	88	3
-- v	11.4000	0.7200	1433	32
-- w	10.0700	1.2300	1187	183
-- Pr	32.4400	1.4710	4393	101
6.75 -- u	0.0393	0.0325	2	0
-- v	0.3290	0.3130	9	8
-- w	0.8100	0.9285	106	138[a]
-- Pr	0.9619	0.8481	94	84
8.25 -- u	0.0930	0.0405	9	2
-- v	0.5440	0.1950	6	21
-- w	1.3460	1.2820	171	162
-- Pr	2.4040	0.8980	183	55

Notes: [a] Denotes that linear shows better results in this particular case

Duct Data

When the approximation techniques were tested on the duct data, it was found that for some cases the linear approximation worked better and for other cases, the spline approximation worked better. It is difficult to be able to say conclusively which method is better. The reason is, that the data itself varies inconsistently. This means that, for small changes in the angle, there were large changes in the data. The angles of the duct at which data was obtained were 18, 24, 30, 48, 60 and 90. Since, extra angles could not be obtained, it was necessary to remove a case, and then approximate through the rest of the cases, checking the approximate value of the removed case against the original value for every point in the data set. The results are shown in table 2. The total number of data points is 18660.

TABLE 2. Duct Approximation Analysis

Angle of the duct & Scalar	Average % Error Linear	Average % Error Splines	Number of noise points Linear	Number of noise points Splines
48 -- Pr	29.710	35.520	6966	8079
-- u	118.640	135.640	6167	7097
-- v	1.164	0.787	36	36
-- w	135.430	156.860	8746	10215
60 -- Pr	14.520	24.220	5358	7338
-- u	120.670	121.430	6394	6491
-- v	1.419	1.084	12	36
-- w	151.000	162.560	8471	9425

It can be seen that the results are not very good for either linear nor splines, although linear seems to fare a little better. This is because the data varies rapidly, and although the curve fitted is smooth, and does not fluctuate much, it still misses the actual values. This is an unavoidable problem as the data rapidly varies, and there is not much way of telling where the removed case is going to occur. This is where the linear interpolation is closer than the spline interpolation. The trend is not possible to predict, when we remove a case from the data set.

The best way to use the approximation tool developed based on this research, would be to run simulations for a few cases, fit the curve, and then run simulations for intermediate cases. Also, care should be taken to see that the same input conditions exist for all cases.

Inaccuracy analysis

As mentioned before; the data for the duct seems to vary rapidly. To demonstrate this, curves were fit through 5 cases, for several sample scalars at single points in the data set. The intermediate sixth case (angle = 48) original results are then plotted against the linear and spline plots Figures 15-18 show the results. The cross mark shows the original solution point, the solid line curve is the spline curve, and the dashed line curve indicates the linear plot solution. The circles indicate the points through which the curve passes close to or passes through.

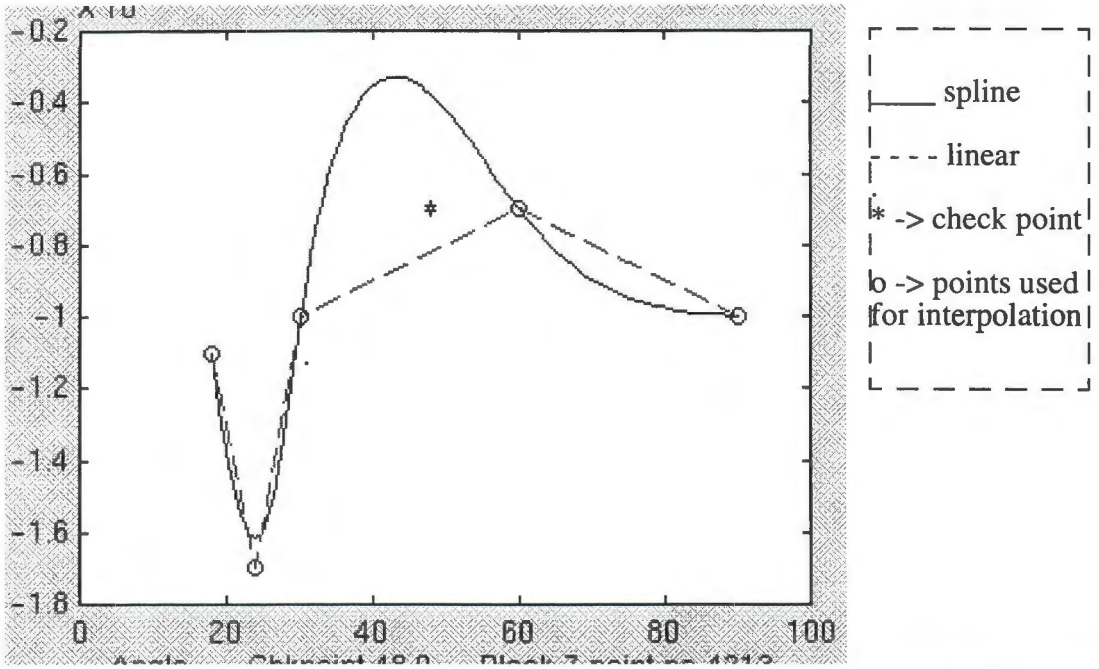

Figure 15. Approximation of pressure over five angles of the duct data set for grid 7, point 4213. Check angle is 48.0.

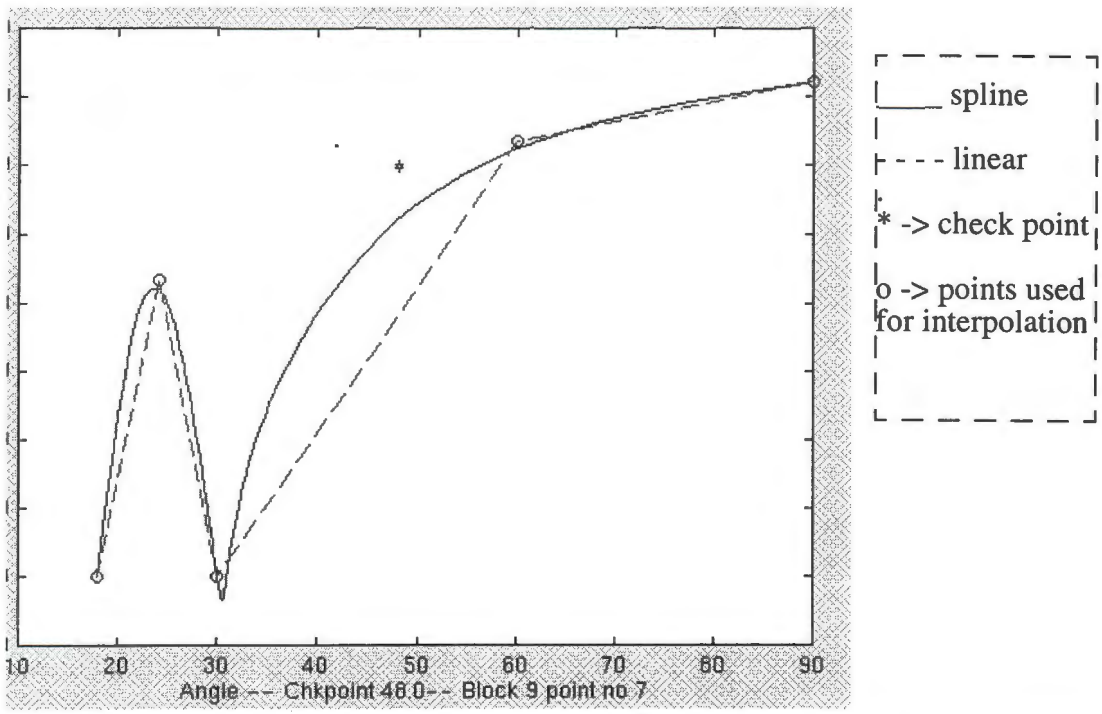

Figure 16. Approximation of pressure over five angles of the duct data set for grid 9, point 7. Check angle is 48.0.(spline closer than linear)

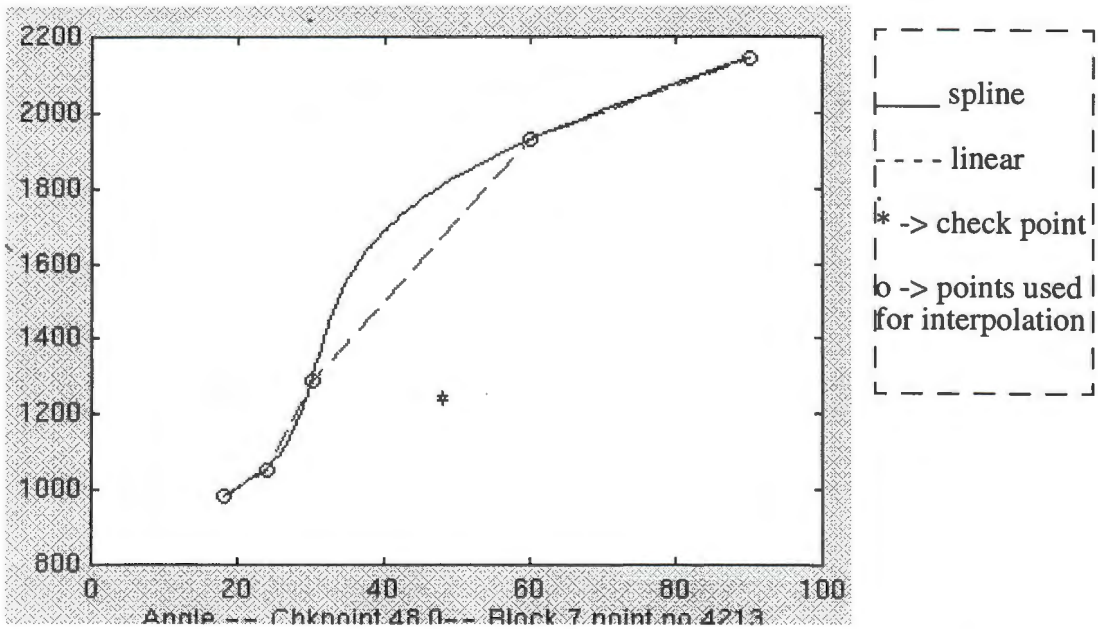

Figure 17. Approximation of X-velocity over five angles of the duct data set for grid 7, point 4213. Check angle is 48.0. (linear closer than spline)

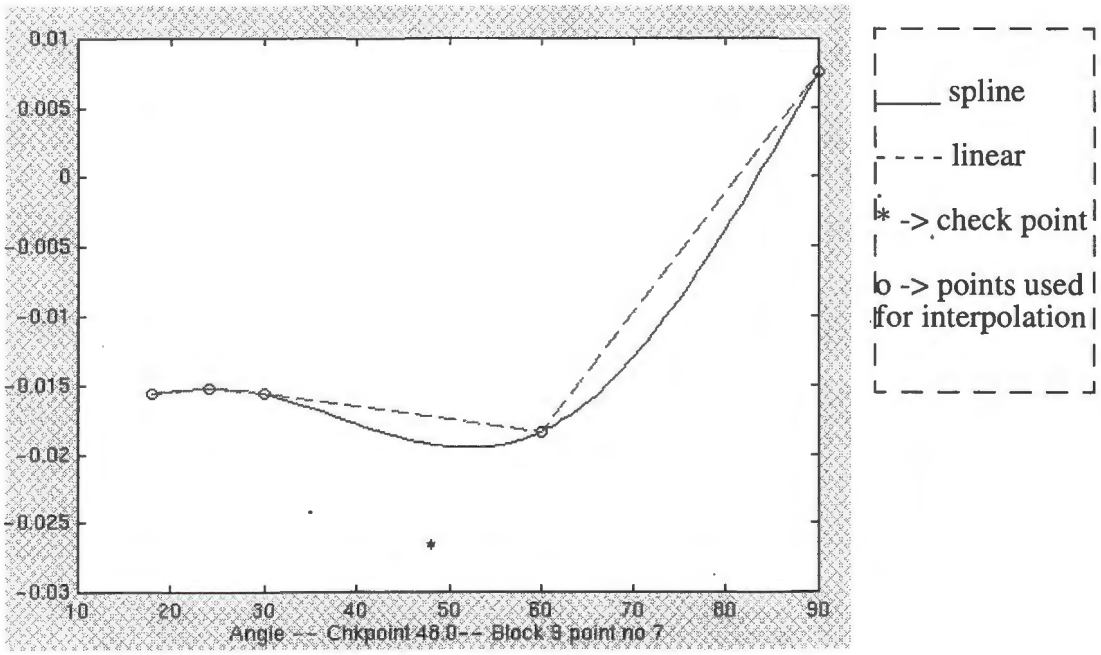

Figure 18. Approximation of pressure over five angles of the duct data set for grid 9, point 7. Check angle is 48.0. (spline closer than linear)

In the cases shown in Figures 15 and 17, the linear approximation is better. For the cases shown in Figures 16 and 18, the spline approximation is better.

Conclusions

Methods of approximation for 3D computational fluid dynamics data were analyzed. The final objective of this research would be the visualization of this data in the virtual environment. In order to achieve that goal, the data approximation needs to be performed interactively in order to maintain high frame rates needed for the display in a virtual environment. With this in mind, ways of approximation were adopted, that would give the best possible results. The data sets used were that of a backward facing step and that of the mixing of two streams from two ducts. The approximations yielded stable results for the backward facing step because of the good behavior of the data and the larger number of approximation cases. Even when the data fluctuated, the virtual knot technique was able to provide good approxi-

mations. In the case of the duct data, however, the fluctuations in the curve were sometimes, too deviant for even this algorithm to handle, especially since the number of cases were few and one of the cases needed to be removed, for checking purposes. If more check points were available, this data also could be stabilized. Generally, when the data fluctuates wildly, a larger number of cases is recommended, both for approximation and checking purposes.

Future work

Turbulent models: One of the main avenues to explore, is to work on better approximations for turbulent models. Researchers at Iowa State University are currently working on a problem involving highly turbulent flow over two cylinders. Instead of linear or spline approximations, they plan to use "close similarity solutions" of the flow to be able to approximate the flow field at various positions of the cylinders.

Artificial Intelligence: In the case of spline interpolation, artificial intelligence methods could be used to determine the number of cases that need to be calculated to get an accurate approximation. Other heuristics could be used to determine the optimum spacing of the solutions.

Other approximation techniques: Other types of approximations, like Bezier, least squares approximation, and some of the fairing techniques described previously. Work could also be done to find out a way of predicting in advance, the best approximation method to use for certain types of data.

Implementation: Finally, the approximation techniques need to be implemented in a fully interactive virtual environment.

References

Hamann, B., and Chen, J., "Data point selection for piecewise linear curve approximation." *Computer Aided Geometric Design,* vol. 11, 1994, 289-301

Hsieh, H. and Chang,W., "Virtual knot technique for curve fitting of rapidly varying data," *Computer Aided Geometric Design*, vol. 11, 1994, pp 71-95.

Kjellander, J. A. P., "Smoothing of cubic parametric splines." *Computer Aided Design*, vol 15 no 3, May 1983, pp 175-183.

Kuroe, Y., Mitsui, M., Kawakami,H., Mori, T., "A Learning Method for Vector Field Approximation using Neural networks", *IEEE International Conference on Neural Networks,* vol. 3, May 1998.

Lott, N. J., Pullin, D. I., "Method for fairing B-spline surfaces." *Computer Aided Design* vol. 20, no. 10, Dec 1998, pp 597-603

Nowacki, H., Lu, Xinmin., "Fairing composite polynomial curves with constraints." *Computer Geometric Design,* vol. 11, 1994, pp 1-15.

Piegl, L. and Tiller,W. *The NURBS Book*, 2nd edition, Springer Verlag 1997, New York.

Poliakoff, J. F., "An improved algorithm for the fairing of non-uniform parametric cubic spline." *Computer Aided Design*, vol. 28, no. 1, 1996, pp 59-66.

Sapidis, N., Farin, G., "Automatic fairing algorithm for B-spline curves." *Computer Aided Design*, vol. 22, no. 2, Mar 1990, pp 121-129.

Schaffer, J. T., Doswell, C. A., "On the Interpolation of a Vector field." *Monthly Weather Review*, 1997, vol 107, pp 458-476

Schroeder, W., Martin, K., and Lorenson, B., *The Visualization toolkit:, An Object Oriented Approach to 3D graphics*, Prentice Hall., New York, 1998.

Shahnawaz, V. J., Vance, J. M., Kutti, S. V, "Visualization of Post Processed CFD data in a virtual environment." *1999 ASME Design Engineering Technical Conference*, Sept. 12-15, 1999, Las Vegas, Nevada.

Zhong, J., Weng, J., Huang, T. S, "Robust and Physically Constrained Interpolation of Fluid Flow Fields.", ICASSP 1992, San Francisco.pp 185-188

Zhong, J., Weng, J., Huang, T. S., "Interpolation of Multidimensional Random Processes with Application to Fluid Flow.", ICASSP 1993. pp 181-184

4. INTERACTIVE APPROXIMATION OF COMPUTATIONAL FLUID DYNAMICS DATA IN A VIRTUAL ENVIRONMENT

A paper to be submitted for the ASME Design Engineering Conference, Sept. 13-16, 2000

Vishant J. Shahnawaz and Judy M. Vance

Abstract

This paper deals with the interactive application of approximation techniques on post processed computational fluid dynamics (CFD) data in a virtual environment. It describes the methodology implemented to allow users to select methods of approximation, parameters for which the response surface needs to be generated, and methods of comparison. For greater interactive ability, methods of parallel programming are implemented, to maintain interactive features during calculations which are detrimental to the immersive capability of the application. The software tools used in developing this application are OpenGL, Visualization Tool-Kit (VTK), and the C2 libraries. This application is implemented for the following hardware, the C2, the HMD, and the desktop monitor in stereo with 3D interaction. This paper builds on our earlier work on the visualization of post-processed CFD data in a virtual environment (Shahnawaz and Vance, 1999).

Introduction

Visualization of CFD data plays an important role in evaluating products that involve fluid flow. Engineers and scientists obtain an understanding of the fluid flow by visualizing flow entities such as streamlines, cutting planes, rakes, and mixing layers. This visualization is performed using CFD solutions which take a large amount of time to generate. If engineers desired to change an input parameter, they need to regenerate the solution, and then read it into either a desktop or virtual environment to visualize it. This not only consumes time but also causes the engineers understanding of the data sets to be disrupted.

Interactive engineering design is one of the areas where virtual reality can be used extensively. Virtual Reality based applications for sensitivity analysis have been developed at Iowa State University by Yeh and Vance (1995). An application for interactive stress analysis of a tractor lift arm was developed by Ryken and Vance (Accepted for publication in 1999). This application was used to approximate the change in stresses with a change in design variables. The type of approximation implemented was linear sensitivity-based approximation. NURBS-based free form deformation was used to change the shape of a connecting rod and the stresses were approximated continuously, hence eliminating the need for any repetition of the FEM analysis Yeh and Vance, (1998).

The approximation methods used are the 'virtual knot' technique (Hsieh and Chang, 1994), and piecewise linear approximation. These approximation methods were found to produce good results when the data did not fluctuate very rapidly. When the data fluctuated rapidly, though, the approximations produced large errors in certain areas. To remedy this situation, a larger number of cases can be calculated and used for the approximation.

Previous work in the field of Virtual Reality-based visualization was done by Bryson and Levit (1992). Bryson, in his work on the virtual windtunnel dealt with the handling of large data sets for the purpose of scientific visualization. This was demonstrated in virtual reality devices such as the BOOM^{TM} and the HMD. Belleman et al. (1998) developed an application for the virtual exploration of fluid diffusion in complex geometries. This application used the CAVE libraries and other software, for a specific purpose of comparing iso-surfaces obtained from flow simulation, to the actual surfaces obtained from CT scan data.

CFD software companies such as FAST (Bancroft et al 1990), Fieldview, Ensight, STAR-CD and other packages have post-processing capabilities for desktop visualization. Some of them such as FAST and EnSight provide for the use of a VR interface. Both packages have the ability of stereo viewing. Shahnawaz and Vance (1999) have extended Bryson's work for display in a CAVE environment. This virtual environment provides an interface that a) encourages collaboration between several participants, b) immerses the participants in the data due to the effects of stereo viewing and wide field of view and c) can be used to present the geometry in real scale.

The common feature of all of these existing applications is that they are used to post process already calculated analysis data. If any changes are desired in the CFD model, a new analysis must be performed and the new data file displayed.The focus of this paper is to incorporate previously developed approximation methods in the virtual environment to facilitate interactive design.

The rest of this paper is organized as follows. The next section provides an outline of the application followed by information on the hardware and software that was used. Next, the input files along with information about the data used is described. This is followed by a discussion of features and methodology of the application which describes techniques of comparison and visualization. Methods of parallel processing to increase interactivity are described next. Finally, conclusions and suggestions for future work are presented.

Overview

This application has the capability of both visualization and approximation. It can read in grid and data files in PLOT3D format. Interaction is through a six degree of freedom joystick called the wand which has three buttons that are used for scaling, translating and rotating the geometry, as well as toggling between interaction modes and interacting with a virtual menu system. The menu system has button options that open, close and switch between menus. Other buttons either toggle interaction modes or activate special functions. The visualization functionality is governed by the same operating principles as in (Shahnawaz et al., 1999). Features exist for the creation, deletion and manipulation of cutting planes, streamlines, rakes, isosurfaces and full field vectors. These features are used to visualize the approximated flow field, and to compare it with the original flow field.

The approximations are calculated based on previous work by Shahnawaz and Vance (1999). Solutions are performed at various values of the selected design variable. Linear approximations or spline interpolation (using the virtual knot technique) are used to obtain solutions to the flow at design variable values for which calculated solutions do not exist. The capability to make a comparison between the approximated solutions and a given intermediate

solution is used initially to guide the user as to which approximation method would be best for a given data set.

Hardware and software

The virtual reality device that runs this application is the C2 virtual reality room. This room has four walls, on which stereo images are projected. The Flock of Birds tracking system tracks the users head and hand position. Users can view the stereo images with CrystalEyes stereo glasses. The computers used to run this device are two SGI Onyx racks, each with twelve R10000 processors and two Infinite Reality graphics pipes. Interaction is through an instrumented wand.

This application was developed using OpenGL, Visualization Toolkit, and the C2 library. This library provides the functions to synchronize the four walls of the C2 and give the user the feeling of being in a single environment. This software is also compatible with devices like the HMD, and the desktop monitor. The Visualization Toolkit library (Schroeder et al 1998) provides functions for the calculation of streamlines and other fluid flow entities and renders the output as OpenGL graphics. OpenGL is used when very few calculations are required, as it performs faster rendering than VTK.

Input files

Before start-up of the application, the input configuration file needs to be setup. The application takes a large variety of input. The input is based on a script file where the user sets up the specifications for the program. The required specifications are as follows.

- Number of grid files *n*
- Grid file names 1 to n
- Number of output data files
- Output data file names 1 to n
- Approximation flag Off/On

- Input parameters for the grid and data files (Which angles are read in)
- Default approximation mode (linear or spline)
- Control point files (For spline approximation)
- Number of vectors and scalar indices corresponding to the components of each vector
- Solution file numbers that correspond to the parameters that are being approximated

Features and approximation methodology

There are two major techniques used here, one is the 'virtual knot technique' (Hseih and Chang, 1994) for the fitting of smooth B-spline curves to the data. This technique helps the curve that has been fit to be smooth and not deviate rapidly, even if the data is fluctuating and is spaced far apart. The other method is piecewise linear approximation, that fits a line between each pair of the solution cases and finds the value at the intermediate parameter in question.

Start-up scheme

The options on the opening menu (see Figure 19) include visualization of any data set, or the choice of modifying the geometry, which is our input parameter in this data set. Every time a new case is activated, the corresponding geometry and data file is loaded.

Figure 19. Menu system

Modification scheme

If the option of modifying geometry is chosen, the user is then directed to a second menu to configure the approximations. The user gets the choice of either linear or spline approximation. There are two options to proceed with.

a) *Analysis*: The Analysis option enables the user to change the geometry to any arbitrary intermediate parameter, and then proceed to calculate and visualize the resulting flow field. Here, the user is given a virtual slider bar which can be used to change the geometry between its first and last configurations. The new geometry is regenerated automatically every time the geometry is changed. The following figure shows the geometry being changed to a different value of the input parameter. The geometry is changed by linear approximation between the grid points of the bounding cases. The bounding cases refer to the two cases between which the modified value lies

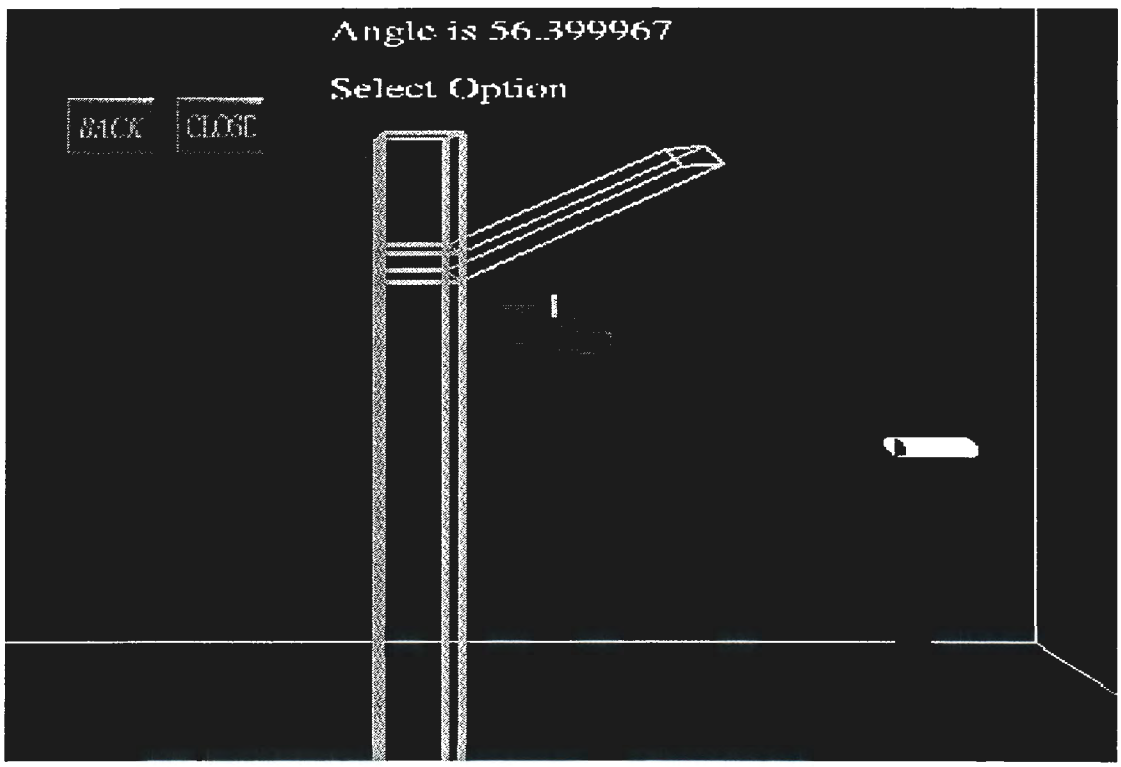

Figure 20. Slider mechanism used to change geometry

b) *Comparison*: This option enables the user to compare an approximated result with an original solution that has been calculated. The user is presented with all the actual solution cases that are available. The user picks which case to approximate. After the approximation is done, the user can then visualize the approximated result, and compare it with the original case. Not all original cases that are available will be used in the approximation. This will give the user an intermediate case for which there is a solution, to test the validity of the approximated results.

Approximation scheme

Once the input variable has been changed, the user can begin to set up the visualization entities. The application has been set up so that the calculations are done only at the time when the user requests a scalar or vector value when creating or modifying an entity. This is because there are a large number of scalars, and the application will take time to generate the whole field for all of them. During this calculation time, which ranges from a few seconds to a few minutes, the application will freeze while it waits for the calculations to finish. This will result in breaking the users sense of immersion and interactivity. Calculating only the desired scalar/vector values instead of the entire output helps to reduce the waiting time. In addition to prevent re-calculation of approximations, a flag is assigned to each scalar, which is switched on, after the calculations are done. This prevents the scalar from being recalculated every time it is requested. The flags are reset every time the input variable is changed.

TIn order to prevent the display from freezing altogether, parallel processing is implemented. A parallel process is spawned off, allowing the user to interact with the flow field and the geometry, while the requested scalars or vector in the flow field is being calculated. The calculation process uses the point inversion technique described in Piegl and Tiller (1997), to calculate the scalar value at every point in the grid. In the case of linear approximation, the scalar value is calculated from the scalar values of the bounding cases. In the case of splines, the scalar value is calculated from the control point files that were read in at the beginning of the application.

On completion of the calculations, the application sends a message to the user, indicating that the calculations are complete. The user may then proceed to put in entities that utilize the calculated scalar or vector. Such entities include cutting planes, streamlines, rakes, isosurfaces, vector fields and all the other features described in Shahnawaz and Vance (1999), to get an understanding of the flow field. Every time the geometry is modified, the flags for all the scalars are reset to the off state, so that recalculation can be done.

Interactive comparison tools

After modifying the geometry, the user can perform comparison between an original solution and the approximate solution. There are two methods of comparison implemented here visual comparison and numerical comparison.

Visual comparison involves the comparison of flow entities from an original solution, to flow entities from the approximation. The user can hence, observe differences between the solution based on the differences between the flow entities.

The color contour map of a scalar on a cutting plane for the approximated solution can be visually compared to that of an original solution. A cutting plane is plane that can be positioned anywhere in the geometry and oriented according to the X, Y or Z axes. This can be either color mapped according to a scalar value, or can have vectors mapped onto it. Features exist to sweep the cutting plane along any of the three axes. The range of the color mapping can be varied using a virtual slider bar. The user starts by creating and positioning a cutting plane in the geometry for the original solution. Then, the user can set up the approximate solution, by picking the comparison option, and the parameter value for the original solution. The entities that have been set up already are preserved, but any new cutting plane will use the approximated data set. The user can then perform the approximation calculations for the same scalar and color map the results onto another cutting plane. The user can then sweep either of the cutting planes back and forth, and watch for differences between the original cutting plane and the approximate cutting plane. (see Figures 21-22).

An isosurface is a surface connecting points of the same scalar value. The user has the capability to set an upper and lower isosurface value, which results in two isosurfaces being

displayed, one for each value. The user can create isosurfaces for a specific pair of values using the original solution, and then proceed to create isosurfaces for the same pair of values using the approximate solution. The disparity in the isosurfaces will give the user an understanding of the disparity between the original and approximate solutions (see Figures 23-24). When Figures 23 and 24 are compared, it can be seen that in Figure 24 the isosurface at the lower part is missing, due to discrepancy in approximations. The user can also super-impose iso-surfaces at the original and approximate values, and observe the discrepancy between them.

.

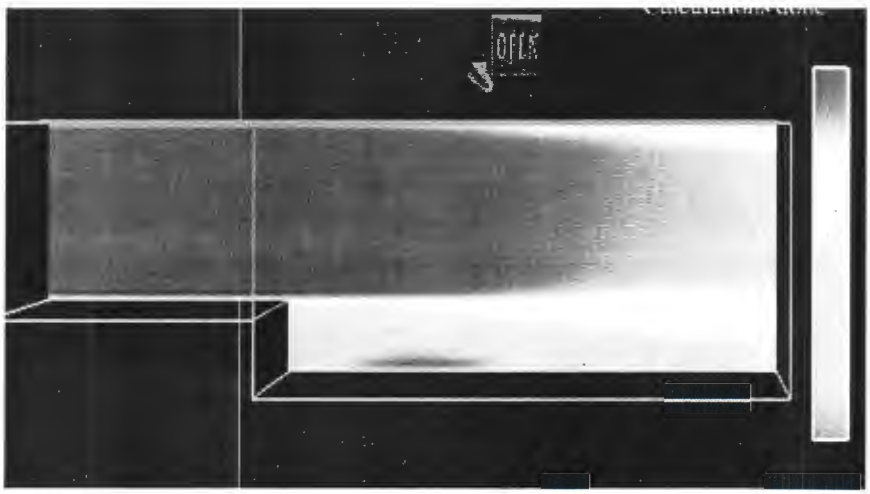

Figure 21. Cutting plane at step length 8.25--original soln---u-vel, max 2.1012 min -0.757

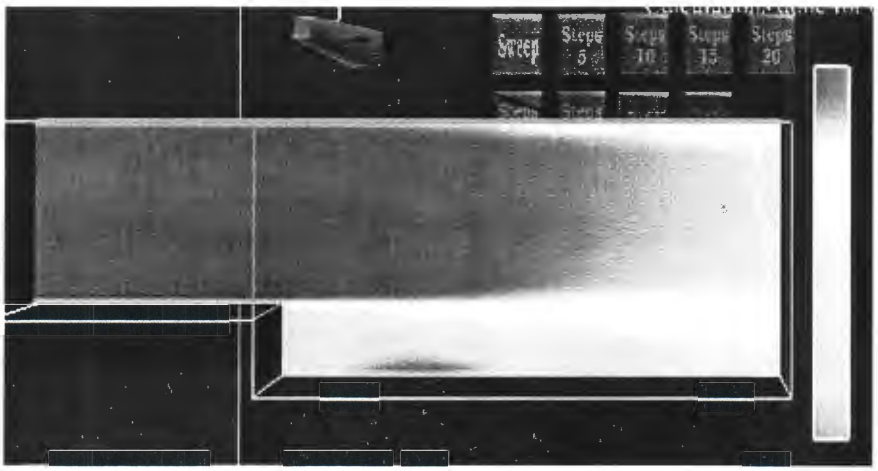

Figure 22. Cutting plane at step length 8.25--app. spline soln--u-vel, max 2.101 min -0.758

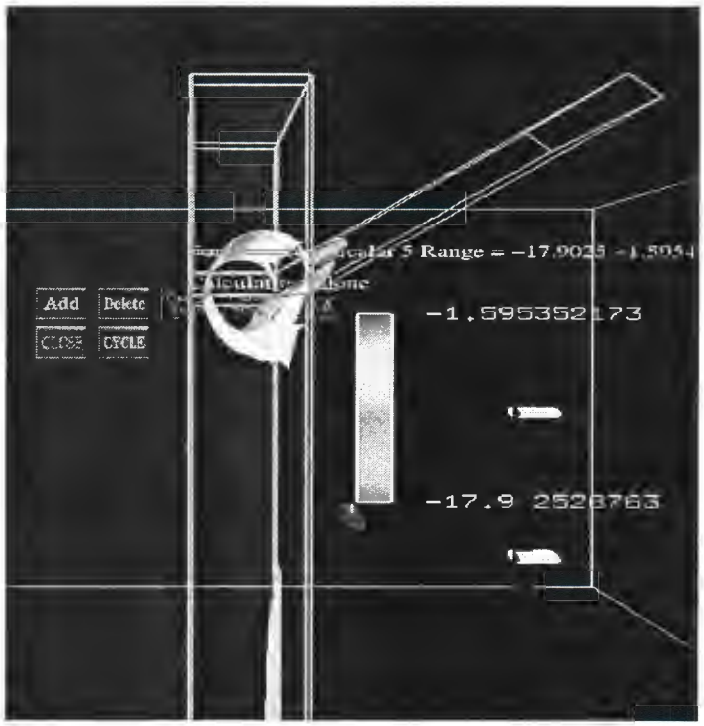

Figure 23. IsoSurface with angle 48 original soln. max -1.5951 min -17.9023

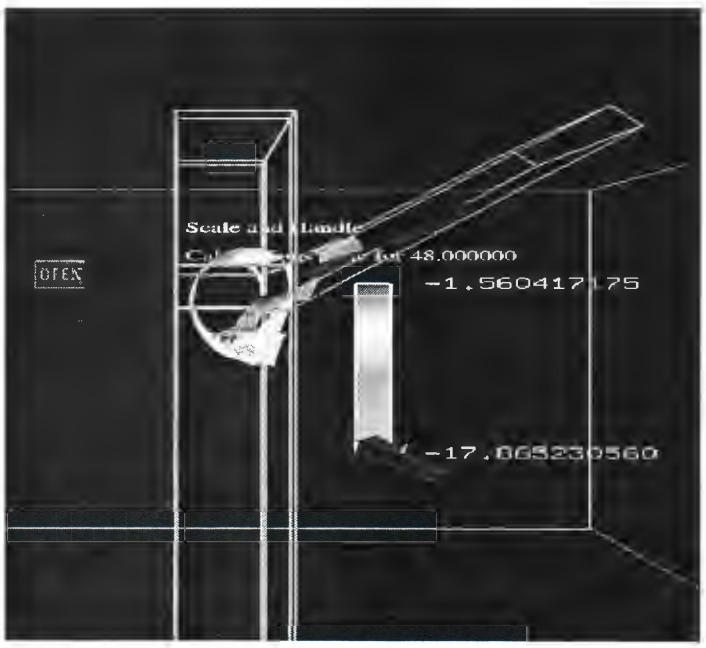

Figure 24. IsoSurface with angle 48 approx soln. max -1.5604 min -17.065.

Streamlines indicate the path that a particle would take, if placed at the "seed" position in the vector field. The users can create multiple streamlines, and integrate them forwards, backwards, or in both directions. The streamlines can also be color mapped according to a scalar and follow a path based on the vector field. The user may superimpose two streamlines, by placing the original solution streamline and the approximate solution streamline, at the same seed point. Then the user can observe the deviation of the approximate streamline from the original one. Also the user may animate particles on the two streamlines. Particle animation gives the user an idea of the velocity field, as the particles will move faster in regions of high velocity and slower in regions of low velocity. The user may observe, if the particles on the two streamlines travel at the same speed or if one of them travels slightly faster than the other.

Figure 29 shows a section of the two streamlines towards the end of the duct. The blue streamline is for the original solution, the red one is for the approximate solution. Both streamlines originate from the same point. The bubbles animated along the streamline show that the particles on the approximate solution have velocity magnitude faster than that of the original solution

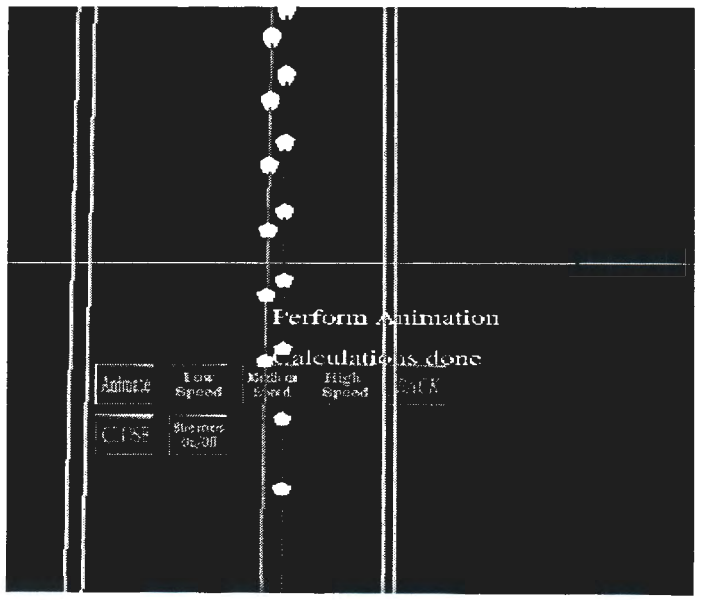

Figure 25. Streamlines originating at the same point for the original solution(red) and the approximate solution (blue)

Numerical comparison was described in detail in our previous work (Shahnawaz and Vance, 2000) on approximation techniques. In this application, the average percentage error is calculated, by calculating the percentage deviation of the approximate solution from the original solution at every grid point and averaging them out over the entire data set. This is done for each scalar or vector that is calculated. The number of "noise points", those grid points that have a deviation greater than 25%, are also calculated. These results can be written to a file for future analysis. Using this information, users can determine which approximation method would be best suited for this data. This technique is not part of the interactive comparative scheme and hence, will not be discussed further.

Parallel programming

Parallel programming is a very useful tool for the purpose of interactive visualization, especially when a large amount of data needs to be computed for the visualization. It becomes even more critical, when the computation needs to be done interactively. In graphics applications, these computations tend to slow down the frame rate, and result in the application "freezing", or locking up until the calculations are complete. This results in a temporary loss of interactive ability and a loss of the sense of immersion. The need to wait for the calculations to complete before performing other operations is also something that needs to be avoided. This is possible through parallel programming, which enables the user to perform operations not related to the parameters that are being calculated. These operations include scaling, rotating and translating the other entities on the screen and the geometry, adjusting and manipulating other entities, etc. Operations that are not possible include creating new entities and changing scalar or vector mapping for existing entities.

Types of parallel programming

There are different ways to implement parallel programming. The type of parallel programming depends on the requirements of the application. In this section, the terminology and issues involved in parallel programming are described.

The types of parallel programming that were tested and utilized in this application are a)Forking processes and b)Multi-threading. Another method of parallel programming that is not discussed here is by using multiple processors, if available on the machine.

Forking processes: Forking of processes can be performed using the UNIX function call, fork(). This function call results in the program creating a child process which has its own copy of the memory, global variables, and local variables that have been declared prior to the call. The operating system schedules time between these two processes, and the other processes of the operating system. The disadvantage to this method is that parallel copies of the memory are being made, and this results in the application failing on smaller systems due to lack of memory, apart from being a waste. (Silbershatz et al., 1998).

Multi-threading: Multi-threading is similar to forking of processes, the difference being that all the processes access the same memory. The thread model consists of two parts as defined in Nichols (1996). One part is called the process and contains the resources used across the entire program such as the program instructions and global data. The other part is called the thread and contains information pertaining to the execution state, such as the program counter and the stack.

The thread model is advantageous only when the program is designed to use multiple threads in the same process. Otherwise, it is the same as any other program. When there is more than one thread, the program can execute more than one procedure at the same time. Each thread can be made to handle a different procedure. Hence, while one process is running all the time, the threads are made to execute different parts of the program while accessing the resources of the process.

Creation of a thread involves the following procedure

- Initializing the thread to certain specifications, or accepting the default specifications
- Specifying a routine where the new thread will begin execution
- Specifying a parameter for the function where the new thread will start.

The Pthreads library, uses the function pthread_create for this purpose. There are other functions that terminate thread execution and that prevent two threads from accessing the same memory at the same time.

A variation to Pthreads, is the sproc() system call on the SGI machines. This call automatically takes care of thread creation and termination. However, it is necessary to ensure that two threads do not access the same memory at the same time. In our application, the sproc() function call is utilized.

Multiple processors: Another effective way of parallel programming is to use multiple processors on a machine that has them. This can be utilized using the MPI library. (Gropp et. al, 1998). It has functions that return the number of processors and assign procedures to different processors. The disadvantage to this method is that it is useful only when there are multiple processors on a machine. If there were 5 to 6 processes running, 5 to 6 processors would be required on the machine.

Inter-process communication

Shared memory is an effective form of inter-process communication. When two processes are running, they modify global variables independently, and need some sort of interface to let one process know what the other is doing, For this purpose, a shared memory buffer is initialized, to which one process writes and another can read. The necessary setup for shared memory is to initialize a structure where the elements of the shared memory are stored. First, it is necessary to create the shared memory and then bind it to the initialized structure. After this is completed, the processes can be forked and can read and write to the shared memory space. (http://goanna.cs.rmit.edu.au/~steveh/tns/solaris/node1.html)

Mutual exclusion

Mutual exclusion is the property that prevents two threads or processes from writing into a section of the memory at the same time. If they did, then one process would overwrite the other, which would yield erroneous results. If the threads or processes are not writing to the same part of the shared space then mutual exclusion is automatically ensured. Otherwise, it is necessary to initialize variables called semaphores that will not allow one process to write to the same space while another process is writing to it. This can be done by setting a continuous while loop that breaks once a process has finished and allows the next process to begin writing.

Results

A virtual reality application for performing interactive approximation of flow fields was developed. The application has features for visualizing the results of an approximated flow field and also comparing it with an original flow field in order to determine which method of approximation would be best. The isosurfaces and cutting planes enabled the user to get an idea of the discrepancy between the original and approximated scalar values in the flow field. The streamlines allowed the user to investigate differences between the original and approximated vector fields.

While performing the calculations, it was found that the user could scale and handle and perform other operations with ease, due to the parallel programming. It was decided to use multi-threading using the sproc() SGI call, so that the application could run on all machines with a main memory size of 500 M and greater. Around 200+ M is needed to accommodate the large amounts of data that need to be read in. since the C2 libraries involve a fork in the process, this results in requiring the system to have at least 500 M of memory. Hence, this application will run ideally in simulator mode on SGI Octanes. In the C2, the application is handled by 2 Onyx SGI racks.

Future work

Scope for future work in this project includes some of the following avenues that could be explored.

Data set volume: Larger and more complex data sets can be read in, with more scalars. Different data sets could be modeled and visualized before or after approximation.

Multiple processors: The application can be used while harnessing multiple processors. Each processor would run its own set of threads.

Point probe: Other means of interaction such as point probes can be utilized. This means that as the users hand moves inside the geometry, the coordinates and current scalar value at the users hand can be displayed.

Conclusions

An interactive application for the approximation and visualization of flow fields with change in input parameters has been developed. Features were provided for interactive visual comparison of original and approximate results. The features provided included cutting planes, isosurfaces and superimposed streamlines. The application was made even more interactive and immersive by using parallel programming through multi-threading. The C2 virtual reality environment facilitated a good three dimensional environment for this approximation and visualization.

References

Belleman, R. G, Kaandorp, J. A., Sloot, P. M. A, "A virtual environment for the exploration of diffusion and flow phenomena in complex geometries," *Future Generation Computer Systems 14*, 1998, pp 209-214.

Bryson, S., Johan, S., Globus, A., Meyer, T., and Schlecht, L., "An Extensible Interactive Framework for the Virtual Windtunnel",*1997 Virtual Reality Annual International Symposium Proceedings*, Alberquerque, NM.

Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, A., Nitzberg, B., Saphir, W., Snir, M., *MPI - The Complete Reference*, The MIT Press, Cambridge, Massachusetts, London, England.

Hsieh, H. and Chang, W., "Virtual knot technique for curve fitting of rapidly varying data," *Computer Aided Geometric Design* vol. 11 1994 pp 71-95.

Nichols, B., Buttlar, D., Farell, J. P, *Pthreads programming*, O'Reilly & Associates, First Edition, 1996.

Piegl, L., Tiller, W., *The NURBS Book*, 2nd Edition, Springer-Verlag 1997.

Ryken, M. J. and Vance, J. M., "Applying Virtual Reality Techniques to the Interactive Stress Analysis of a Tractor Lift Arm." *Finite Elements in Analysis and Design*, (To be published, 1999)

Shahnawaz, V. J., Vance, J. M., Kutti, S. V., "Visualization of Post Processed CFD Data in a Virtual Environment." *Proceedings of DETC '99, 1999 ASME Design Engineerinf Technical Conferences*, September 12-15, Las Vegas Nevada.

Shahnawaz, V. J., Vance, J. M, "Approximation of Computational Fluid Dynamics data for use in virtual reality." submitted to the 2000 ASME Design Automation Conference.

Schroeder, W., Martin, K. and Lorensen, B., *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Prentice-Hall, Inc., New York 1998.

Silberschatz, A., and Galvin,P.B, *Operating System Concepts*, Addison Wesley Longman, Inc, 1998.

Yeh, T-P. and Vance, J. M, "Combining MSC Nastran, sensitivity methods and virtual reality to facilitate engineering design," *Finite Elements in Analysis and Design*, vol 26, 1997, pp 161-169.

Yeh, T-P. and Vance, J. M., "Applying Virtual Reality Techniques to Sensitivity based Structural Shape. S Design", *Journal of Mechanical Design*, vol 120, December 1998.

5. CONCLUSIONS

General Discussion

An alternative approach to visualizing post processed CFD data has been developed here. This approach utilized the advantages of different immersive virtual environments, thereby allowing the user to interactively explore and investigate the flow. User feedback was obtained regarding the different VR devices used. Three interface devices, namely, the Head Mounted display (HMD), the C2, and the desktop monitor in stereo, were used. It was found that the C2 was the best interface device for this application, as it provided the most immersive and comfortable interactive environment. The HMD was found to be obtrusive, and could only be used by one user at a time. The stereo display showed problems of ghosting, and double images, along with infeasible interaction.

An interactive method for investigating design change effects on CFD results has been developed. The approach involved approximating between CFD data sets which were generated at different values of an input parameter such as geometry. Two basic approximation techniques were analyzed for accuracy by re-generating an approximate intermediate data set and comparing the result with a real time result that was generated at that value. It was found that in some cases when the data sets are spaced unevenly, and far apart, the corresponding errors were large. The application was tested on two data sets, a backward facing step and a forked duct with mixing streams and the error analysis was performed. In general a larger number of approximation cases is needed when the flow characteristics fluctuate wildly.

Interactive comparison methodologies were implemented, thereby enabling the user to visually compare an approximated and original data set, and observe the differences between the two. These visual cues are in the form of cutting planes, streamlines, or isosurfaces.

The sponsors of this research tested these approaches, and succeeded in re-confirming their understanding of the fluid flow and noticed flow phenomena that were not seen before. They were able to verify inadequacies in their modeling of the flow by visualizing it in the virtual environment.

The results obtained from the duct with mixing streams showed that large CFD numerical simulations needed to be done on only a few cases. The models could be interactively tested by modifying the input parameter to any desired value in the virtual environment and performing the approximations to regenerate the new solution. The new solution was visualized and gave the users a good approximate idea of what to expect of the flow at that value of the input parameter.

Recommendations for future research

There are many avenues for future research in this area. They are elaborated below.

Generality: The generality of this application as a post-processing tool can be further improved to be able to specify a larger number of data formats, and to take in a larger number of scalars and vectors. Presently only PLOT3D files can be input.

Computation speed: This is a key issue in CFD post-processing. CFD grids can easily become very large and trade-offs must be made between displaying all of the data and maintaining real time display speed for VR on the order of 12-15 frames per second as a minimum. So far a maximum of 50,000 cells has been read in. When the full velocity field was displayed, the frame rate dropped below 10 frames per second. Displaying multiple rakes with a cutting plane also reduced the frame rate. Researchers at the National Center for Supercomputing Applications are developing an interface where VTK and Iris Performer are combined in an effort to increase display performance (Leigh et al., 1998). Bryson and Gerald-Yamasaki (1992) have investigated implementation of a distributed architecture for the Virtual Windtunnel. Both of these approaches will be investigated in the near future.

Calculation of new scalar and vector properties: This capability will allow users the ability to recalculate more parameters by using the existing scalar and vector data.

Multiple usage and VR networking: This development will enable different users in different locations to interact with the application at the same time. At present, the application can only run in one location. In the future, several users at different locations in different parts of the world will be able to work with the same data in the same virtual environment.

Menu system: Currently, the menus are stationary. At times it would be useful to move the menus out of the way. There are many different menu paradigms possible in VR. Different options such as the VUI developed by Daniel Heath (1998) will be investigated.

Data set volume: The application could be modified to allow larger and more complex data sets to be input along with more scalars. Presently, when large data sets are visualized, the frame rate drops. Different data sets could be modeled and visualized before or after approximation. Parallel processing using multiple processors could enable larger data sets to be visualized and approximated interactively.

Approximation methods: One of the main avenues to explore, is to work with more turbulent models in approximation. A research group at Iowa State University are currently working on highly turbulent flow over two cylinders. Instead of linear or spline approximations, they plan to use "close similarity solutions" of the flow to be able to approximate the flow field at various positions of the cylinders.

Artificial intelligence: In the case of spline interpolation, artificial intelligence methods could be used to find out the number of cases that need to be calculated to get an accurate approximation. Other heuristics could be implemented to determine the spacing of the solutions.

Other approximation techniques: Other types of approximations like Bezier or least squares approximation (Piegl and Tiller, 1997)could be utilized and checked for accuracy. Work could also be performed to find out a way of predicting in advance which method would give the best approximation method to use for certain types of data.

Finite Element data: The approximation techniques described here could be extended to apply to finite element models. Preliminary work in this area has been performed by Yeh and Vance (1998).

References

Bryson, S. and Levit, C., "The Virtual Wind Tunnel: An environment for the exploration of three dimensional unsteady flows", *IEEE Computer Graphics and Applications*, v. 12, July 92, pp. 25-34.

Heath, D. J., "Virtual User Interface (VUI) A windowing system for VR", *Second International Immersive Projection Technology Workshop Proceedings*, Iowa State University, Ames, IA, May 11-12, 1998.

Leigh, J., Rajlich, P. J., Stein, R. J., Johnson, A. E., and DeFanti, T. A., "LIMBO/VTK: A tool for rapid tele-immersive visualization," *IEEE Visualization '98 Proceedings*, October 18-23, 1998, Research Triangle Park, NC.

Piegl,L. and Tiller,W. "The NURBS Book," 2nd edition, Springer Verlag 1997.

Shahnawaz, V. J., Vance, J. M, "Approximation of Computational Fluid Dynamics data for use in virtual reality." submitted to the 2000 ASME Design Automation Conference.

Yeh, T.-P., and Vance, J. M., "Applying Virtual Reality Techniques to Sensitivity-based Structural Shape Design," *Journal of Mechanical Design*, vol. 120, December 1998, pp. 612 - 619.

ACKNOWLEDGEMENTS

I wish to thank, first and foremost, my professor, Dr. Judy Vance who was to me, a friend, teacher, and guide. It was through her support and guidance, that this entire work came into being. I would also like express my thanks to my committee members, Dr. James Bernard and Dr. Leslie Miller. My sincere thanks to the Procter and Gamble Company, Cincinnati, Ohio, for their contributions during the past two years. Last, but not the least, I would like to appreciate, all the help that I received, from the staff, faculty and fellow students at the Virtual Reality Applications Center, Ames, Iowa.