◎ 논 문

객체지향 개념을 반영한 유동해석 후처리 프로그램에 대한 연구

나 정 수*1, 김 기 영*2, 김 병 수*3

Study on a post-processing program for flow analysis based on the object-oriented programming concept

J. S. Na, K. Y. Kim and B. S. Kim

In the present study, a post-processing program is developed for 3D data visualization and analysis. Because the graphical user interface(GUI) of the program is based on Qt-library while all the graphic rendering is performed with OpenGL library, the program runs on not only MS Windows but also UNIX and Linux systems without modifying source code. The structure of the program is designed according to the object-oriented programming(OOP) concept so that it has extensibility, reusability, and easiness compared to those by procedural programming. The program is organized as modules by classes, and these classes are made to function through inheritance and cooperation which is an important and valuable concept of object-oriented programming. The major functions realized so far which include mesh plot, contour plot, vector plot, streamline plot, and boundary plot are demonstrated and the relevant algorithms are described.

Key Words: 후처리(post-processing), 데이터 가시화(data visualization), 유동해석(flow analysis), 객체지향(object-oriented programming, OOP), 클래스(class), 동적 메모리(dynamic allocation)

1. 서 론

유동현상을 수치적으로 해석하는 전산유체역학(CFD, Computational Fluid Dynamics)은 해석하고자 하는 대상의 형상에 대하여 격자를 구성하는 전처리 과정(Pre-processing), 유동의 지배방정식을 적절한 초기조건 및 경계조건과 함께 계산하여 유동현상을 해석하는 과정(Main-processing), 그리고 계산된 수치적인 결과를 이해하고 분석할 수 있게 해주는 후처리 과정(Post-processing)을 거친다. 이러한 3가지 과정 중 전처리 과정과 유동계산 분야는 많은 연구를 통하여 국내에서도 많은 경험과 연구 자료가 축적되어 있고, 이를 바탕으로 우리나라 자체 기술로 비행체를 설계/제작하고 있다. 그러나 후처리 분야는 상대적으로 국내에서 연구가 많이 진행되어 있지 않은 실정이기 때문에 고가의 외국 상용 프로그램을 구입하여 사용하고 있다.

CFD 계산 결과에 대한 3차원 가시화(3D visualization) 프로그램은 80년대 중반에 NASA Ames의 PLOT3D로부터 시작하였다[1]. 그 당시의 PLOT3D는 정렬격자형태의 데이터만을 처리 할 수 있도록 프로그램 되었는데, 그 후 MIT에서 비정렬 격자 데이터 가시화를 위한 알고리즘을 개발하기 시작하였다. 점차 CFD 계산이 정상 상태 계산뿐만 아니라 비정상 상태를 계산하면서 비정상 유동해석 데

* 2004년 2월 2일 접수
*1 학생회원, 충남대학교 대학원 항공우주공학과
*2 학생회원, 충남대학교 대학원 항공우주공학과
*3 정회원, 충남대학교 항공우주공학과(kbskbs@cnu.ac.kr)

이터에 대한 후처리 및 가시화연구가 시작되었다. 유동해석 분야가 다양화되면서 그에 따른 후처리 도구들이 필요하게 되어 여러 갈래의 가시화 프로그램들이 나타났고 각각의 특색을 가지고 발전하게 되었다. 그리하여 현재는 Tecplot, Ensight, AVS 등의 상용 프로그램들이 활용되고 있으며 CFD의 발전 및 CFD를 이용한 산업 발전에 중요한 역할을 하고 있다.

최근의 경향은 CFD 코드들 자체에 후처리 기능을 포함하고 있는 경우도 있으며[2], CFD의 세 과정이 통합되어 하나의 큰 프로그램으로 되어가고 있다. 계산환경에서도 전 세계적으로 인터넷상의 네트웍 자원들을 모아 마치 슈퍼컴퓨터처럼 사용하고자 하는 그리드 프로젝트가 활발하게 연구되고 있으며 국내에서도 한국과학기술정보연구원(KISTI)을 중심으로 최근에 연구를 시작하였다.[3] 그리드 프로젝트는 고성능 계산 응용 분야 연구자들의 참여가 중요한 비중을 차지하고 있고,, 그 중에서도 CFD 분야는 아직도 강력한 컴퓨팅 파워를 필요로 하고 있기 때문에 이러한 그리드 환경은 CFD 문제를 적용해 볼 수 있는 좋은 예가 되고 있다. 이러한 환경에서 다자간 연구를 하고 서로의 연구정보를 공유하고 이해하기 위해서는 계산 환경뿐만 아니라 실제적인 가시화 도구가 필수적이라 할 수 있다. 그리드에서 CFD 계산 환경의 예 중 하나로 Cactus[4]를 들 수 있는데, 계산뿐만 아니라 후처리 기능도 다양하게 지원하려는 목표를 가지고 발전되고 있고 현재 기초적인 원격 제어(remote control)와 원격 가시화(remote visualization) 기능을 갖추고 있다. 그러나 CFD 데이터에 대해서 일반적인 가시화 기능을 수행하기에는 부족한 점이 많이 있기 때문에 이에 따른 프로그램 개발이 계획되고 있다. 따라서 국내에서도 이러한 시대 흐름에 발맞추어 그리드 환경에서 활용 가능한 프로그램 개발을 위한 자체 후처리프로그램 개발 능력 확보가 필요하다.

국내에서 후처리 프로그램들은 몇몇 연구자들에 의해서 개발되고는 있지만[5,6] 상용프로그램 수준의 기능을 구현하지는 못하고 있고 또한 일반적으로 지속적인 성장을 못하고 있다. 그 이유로는 개발 경험이 부족한 이유도 있겠지만 다중 블록 정렬격자(multi-block structured grid)와 비정렬격자(unstructured grid), 그리고 각각에 대한 2D 와 3D 데이터, 그리고 사용자를 위한 다양한 기능들이 기존의 순차적인 프로그램 방식으로 프로그램 하기에는 너무나 다양하고 방대하기 때문이다. 객체지향프로그램(OOP, object-oriented programming)은 이러한 대규모의 프로젝트를 통합성과 일관성을 유지하면서 프로그램을 할 수 있도록 개발되었으며 현재 대부분의 상용프로그램들은 OOP를 기반으로 하여 프로그램 되고 있다.

본 연구에서는 이러한 배경에 의하여 다음과 같은 몇 가지 중점 사항을 고려하여 프로그램을 작성하고 있다. 첫째 외국 상용 소프트웨어는 다양한 분야에서 일반적으로 사용 할 수 있도록 하기 위하여 아주 많은 설정과 기능들이 있어서 오히려 복잡한 느낌을 주며 고가로 팔리고 있다. 따라서 우리 실정에 맞도록 많이 사용되는 기능을 편리하게 만드는 것은 경쟁력이 될 수 있다. 둘째 잘 구성된 GUI(graphical user interface)는 사용자에게 친숙한 환경을 제공해주어 보다 많은 사용자들을 확보하고 프로그램을 성장시킬 수 있는 기본적이면서 중요한 사항이다. 셋째 현재의 프로그램이 구현하는 기능뿐만 아니라 지속적인 성장을 하기 위해서는 OOP 개념을 기반으로 한 구조설계가 되어야 한다. 마지막으로 기존의 검증된 가시화 알고리즘을 적용하여 적합한 결과가 나와야 한다.

본 논문에서는 초기 버젼에 해당하는 유동해석 후처리프로그램 DAVA1.0(DAta Visualization and Analysis)[6]을 개발했던 경험을 바탕으로 DAVA 2.0을 개발하면서 얻어진 연구 자료를 제시하고자 한다.

2. 프로그램의 구조 설계

2.1 개발 목표

본 프로그램의 궁극적인 개발 목표는 다음과 같다.

▶ 특화된 가시화 프로그램

본 프로그램의 개발 목적은 외국 상용 소프트웨어들의 가시화 기능들 중에서 많이 쓰이면서도 중요한 기능을 구현하여 데이터 가시화 및 분석용으로 활용하고, 특정 문제에 대한 특화된 프로그램으로 발전시키는 것에 있다. 외국 상용 소프트웨어는 일반적으로 아주 많은 설정과 기능들로 인하여 사용방법을 익히는데 많은 시간이 소요되고, 또한 프로그램 사용법 교육을 받아야하는 경우도 있다. 이러한 상황을 외화 낭비와 기술 종속이라는 측면에서

고려해 볼 필요가 있다. 특화된 프로그램의 개발 방향으로는 가장 많이 사용하는 기능들을 편리하게 구현하여 사용 방법을 익히는데 드는 시간을 줄여주는 방법, 기계 계열과 항공 우주 계열의 특징에 따른 효과적인 분석기능 구현, Unix/Linux, Windows 등에서 모두 실행 가능한 플랫폼 독립성, 그리드 환경에서 활용 될 수 있는 프로그램으로의 발전 등을 들 수 있다.

▶ 편리한 GUI 구성

잘 구성된 GUI는 사용자에게 친숙한 환경을 제공해주어 user 들을 확보하고 프로그램을 성장시킬 수 있는 가장 기본적이면서 중요한 사항이다. 이를 위하여 DAVA 2.0에서는 버전 1.0 개발경험을 바탕으로 전체적인 프로그램 구조설계 및 GUI 디자인을 먼저 하고 그에 따른 프로그램을 작성하는 방식으로 진행해 나갔다. 데이터 분석 기능들 중에서 사용빈도 수가 많은 기능을 중심으로 GUI 구성을 하였다. 또한 상용 프로그램들이 구현하고 있는 편리한 3D 그래픽 컨트롤을 위해서 마우스와 키보드를 통하여 3D 데이터를 확대/축소, 평행이동, 회전 등이 구현되도록 하였다.

▶ OOP 개념에 근거한 프로그램 구조 설계

CFD 계산과정에는 크게 정렬격자와 비 정렬격자를 이용한 방법들로 나뉘어 질 수 있는데 각각에 대해서 2D와 3D 데이터가 있다. 정렬격자는 복잡한 형상에 대해서 다중블록(multi-block)으로 나누어 처리 할 수 있다. 그리고 복잡한 형상에 대한 용이한 격자 생성 및 상대운동에 대한 모사를 위하여 중첩격자를 사용하는 경우도 있다. 비정렬 격자는 2D일 경우 각 cell은 삼각형이나 사각형 그리고 오각형 등으로 구성될 수 있고 3D일 경우 사면체, 피라미드, 프리즘, 육면체 등등으로 이루어 질 수 있다. 이러한 다양한 데이터에 대해서 C 언어 프로그래밍과 같은 순차적 방법으로 프로그램 할 경우 너무 많은 경우의 수를 처리해야 하고 프로그램 코드가 구조화 되지 못하고 점점 복잡해져가는 문제에 다다르게 된다. 최근의 대규모의 프로젝트들이 이러한 문제를 OOP 개념을 도입하여 해결하면서 확장해나가고 있는데 이는 큰 프로그램을 연관성 있는 부분들로 나누고 조직화 함으로써 확장성 및 관리의 용이성이 좋기 때문이다. DAVA 2.0에서는 OOP 개념에 근거하여 다중블록 정렬격자와 비정렬격자 형태의 데이터를 처리 할 수 있도록 구조 설계를 하였고 지속적으로 통합 환경에 적합한 구조를 유지해 나아갈 계획이다.

▶ 가시화 알고리즘 적용

후처리 프로그램은 80년대 중반부터 시작하여 급속히 발전하였고 이에 대한 연구자료 및 논문이 축적되어있다. 따라서 본 프로그램의 가시화 기능 구현을 위해서 기존의 검증된 가시화 알고리즘을 적용하였으며, 참고할 만한 기존 알고리즘을 찾지 못한 경우는 자체 개발하여 진행하였다. 스칼라(scalar)형 데이터 가시화 기능으로는 contour plot, iso-surface plot 등이 있고, 벡터(vector) 데이터 가시화 기능에는 vector plot, streamline plot, streamribbon, streamtube등이 있다.

2.2 프로그램의 순서도

가시화 프로그램이 실행되는 기본 절차는 아래의 Fig. 1과 같다. 데이터를 읽은 후 데이터를 정렬격자와 비정렬격자로 구분하여 메모리를 동적으로 생성하고 클래스화 된 구조로 저장한다. Qt[7]에서 지원하는 GUI를 사용하여 Event call 하면 가시화 알고리즘을 데이터에 적용하여 계산 후 기능을 수행하고 필요한 정보를 나타내준다.

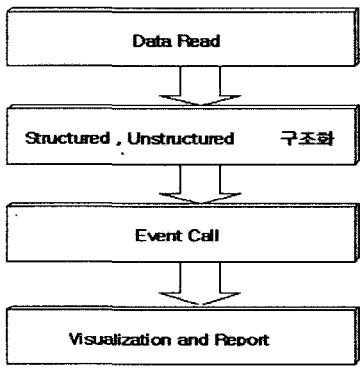


Fig. 1 순서도

2.3 클래스 협력 도표

객체지향 프로그램을 자동적으로 문서화 해주는 doxygen을 이용하여 클래스 협력 도표를 만들면 Fig. 2와 같은 도표를 얻을 수 있다. 여기서 각각의 박스들은 클래스를 의미하고 실선은 상속관계를, 그리고 점선은 협력관계를 나타내고 있다. 클래스 협력 도표에 나온 중요 클래스들의 주요 기능을 살펴보면 다음과 같다.

Point 클래스에서는 계산결과인 종속변수를 그 수에 따라 동적으로 메모리 할당을 하고 이를 PointSet 클래스에서 다시 격자수에 해당하는 Point 클래스의 객체를 동적으로 생성하게 된다. 즉 PointSet 클래스는 Point 클래스 객체들의 집합을 가지고 있으며 이에 대한 데이터의 min, max 값과 같이 정렬격자와 비정렬격자에 모두 적용될 수 있는 속성과 기능으로 구성되어 있다. 이를 상속받아 크게 정렬격자와 비 정렬격자로 나누고 각각에 정의된 기능이 수행된다. Reader 클래스는 GL3DviewBase 클래스에서 넘어온 정렬격자와 비정렬격자의 객체 포인터를 이용하여 데이터 종류가 어떤 것인지를 판단하는 일과 그 데이터 종류에 따라 데이터를 로드하는 작업을 담당한다. 정렬격자인 경우 MultiStructuredGrid 클래스에서 단일블록들의 배열로써 다중블록을 처리 할 수 있도록 되어 있는데, 이곳에서 다양한 세부 기능을 설정 할 수 있도록 되어 있다. Gridgen 클래스에서는 격자 및 데이터를 생성 하는 클래스로 사용되고 있는데 추후 격자 생성용 클래스로 확장할 계획이다. 비정렬격자는 Tetra, Pyramid, Hexahedron 클래스 형태의 객체들을 사용하는데, 이 클래스들은 가상함수가 구현되어 있는 Cell 클래스로부터 상속을 받게 되며 이렇게 함으로서 다형성을 지원할 수 있다. 이러한 다형성의 장점은 상속받는 객체들을 하나로 표현함으로써 프로그램이 구조화되고 간단해진다는 것이다. 이러한 클래스들은 Qt 라이브러리를 이용하여 GUI를 구성하고 있는 DAVAMainFrame 클래스를 통해 수행 명령을 주고받는다.

2.4 GL3DviewBase 클래스

지면 관계상 여러 클래스들 중 전체적인 3D 그래픽 처리를 관리하는 GL3DviewBase 클래스 수행 기능에 대해서만 살펴보면 다음과 같다. DAVAMainFrame 클래스에서 받아온 가시화 기능 설정 명령들을 Qt에서 지원하는 슬롯(Slot)을 통하여 GL3DviewBase 클래스에 가시화 선택 모드를 설정한다. Fig. 3에 나와 있는 것과 같이 설정 값에 따라 3차원 그래픽 화면을 컨트롤하거나 정렬격자와 비정렬격자 등에 세부 설정 값이 적용되도록 되어있다. GL3DviewBase 클래스는 사용자로부터 데이터 입력 명령이 들어오면 Reader 클래스를 생성 후 이 곳으로 정렬격자와 비정렬격자 클래스의 객체를 넘겨주고 이 객체에 데이터를 로드한다. 파일을 읽는 과정이 완료 되면 파일에 대한 정보를 각 클래스에 넘겨주고 Reader 클래스 자신은 소멸함으로써 낭비되는 메모리를 줄이고 있다. 이러한 과정이 완료되면 Qt OpenGL에서 제공하는 paintGL() 멤버 함수를 호출하여 가시화 기능을 수행한다. 여기서 가시화 기능은 Fig. 3에 나와 있듯이 설정 값에 따라 각각의 세부 클래스에 구현되어 있는 멤버함수를 호출하여 실행하도록 되어 있다. 즉, 이 클래스가 전체적인 그래픽처리를 관리하

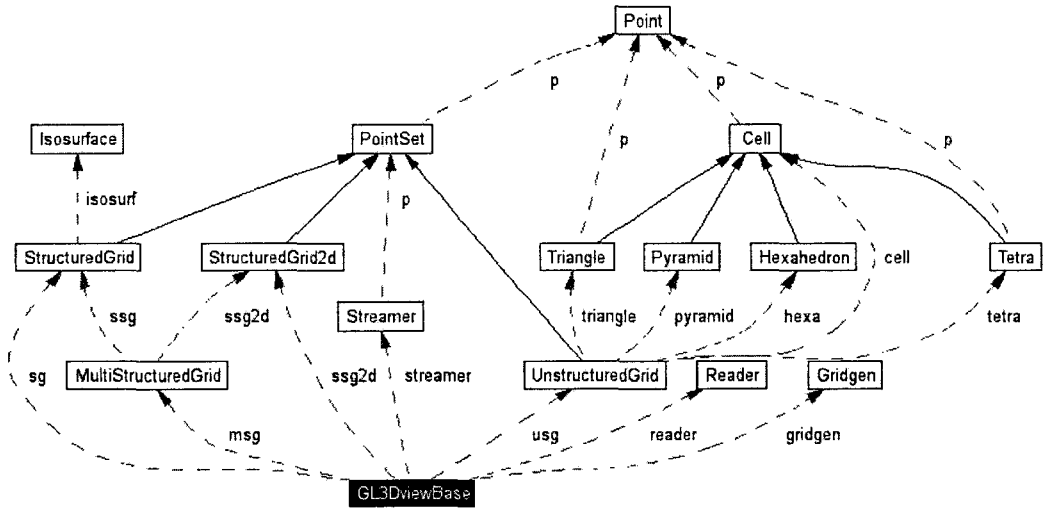


Fig. 2 클래스 협력 도표

고 세부 기능들은 하부 클래스에서 담당하도록 함으로써 방대한 소스 코드를 모듈화 할 수 있는데 이것은 객체지향의 캡슐화 개념이 반영된 대표적인 예이다.

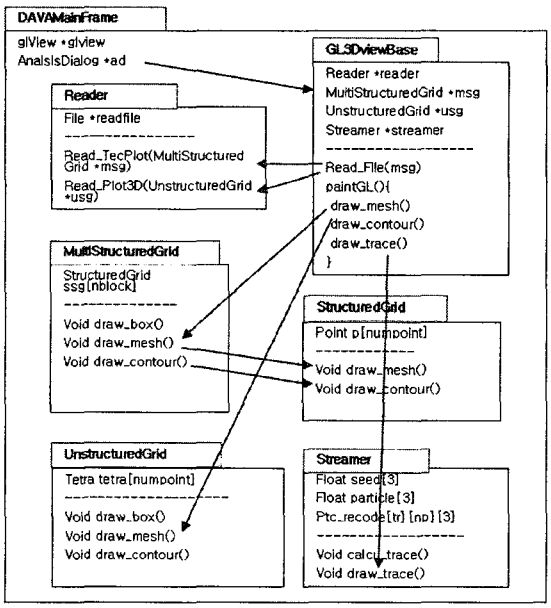


Fig. 3 DAVA 클래스 내부 구성도

3. 가시화 알고리즘

3.1 스칼라 방법

3.1.1 I, J, K surface contour

Contour를 그리기 위한 방법으로 많이 쓰이고 있는 방법은 marching square 알고리즘이다[8]. 이 알고리즘의 원리는 사변형 형태의 모든 셀에 대하여 각 셀의 변(edge)에서 동일한 값을 꼭지점으로부터 보간하여 찾아내고 선으로 이어 주는 것이다. 하나의 셀에서 contour-line을 만들 수 있는 경우는 Fig. 4에서처럼 모두 16 가지이지만 대칭성을 고려하면 Fig. 5에서와 같이 네 가지로 줄일 수 있다. 한 예로서 Fig. 4의 Case 1과 Case 2의 경우처럼 회전을 통해서 같은 사변형으로 변환될 수 있는 모든 경우들은 꼭지점 중 하나를 절단하는 하나의 선분을 갖는다. 또 다른 대칭성은 Case 0과 Case 15번의 경우처럼, 검은색과 흰색을 바꿈으로써 같아질 수 있는 경우들 사이에서의 대칭성이다. 따라서 이들 네 가지의 경우에 대해서 프로그래밍을 해줌으로써 모든 경우의 수를 처리할 수 있다.

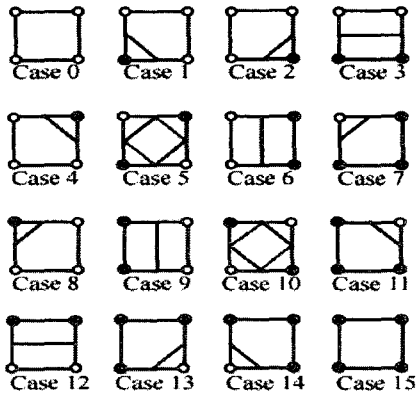


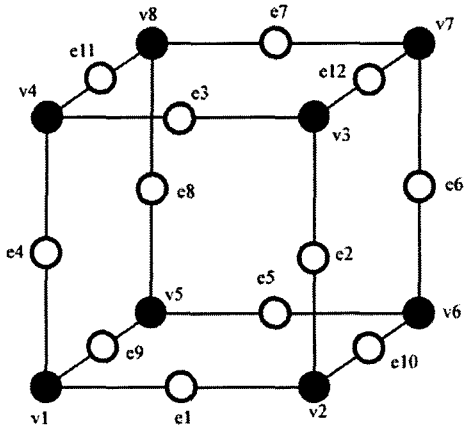
Fig. 4 Marching square의 16가지 경우

Fig. 5 Marching square의 최종 4가지 경우

3.1.2 Iso-surface

Iso-surface를 구현하기 위한 알고리즘으로 널리 사용되고 있는 방법은 marching cubes이다[8]. 이 방법은 삼각형 조각(patch)을 이어 붙여 면을 만드는 방법으로써 Fig. 6에서처럼 육면체의 꼭지점의 선택여부에 따라 2^8 = 256가지의 경우의 수가 생길 수 있지만, contour에서와 비슷하게 대칭성을 고려하면 Fig. 7의 14가지 경우로 축약시킬 수 있다. 따라서 이들에 대해 리스트를 작성하고 요건에 맞는 경우를 그려주면 marching cube를 구현할 수 있다. 예를 들어, v1, v3, v4 가 선택되었다면 $2^0+2^2+2^3=13$을 이용하여 13번째 리스트의 경우를 디스플레이 해준다. 이 알고리즘을 다시 정리 하면 다음과 같다.

1. Cell을 선택
2. Cell의 각 vertex의 내/외부 상태를 계산
3. Edge lists를 생성
4. Edge table에서 cell의 상태를 찾아 읽기
5. Table에 있는 각각의 edge에서 두 꼭지점의 가중치를 선형 보간하여 edge에서의 좌표 찾기



Case = v8|v7|v6|v5|v4|v3|v2|v1

Fig. 6 Marching cube

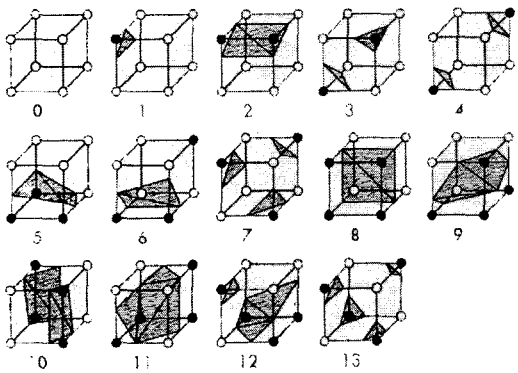


Fig. 7 Marching cube의 14가지의 경우

3.2 벡터 방법

3.2.1 Streamline

벡터들의 접선을 따라 이은 선, 즉 스트림라인의 구현 절차는 먼저 데이터의 물리적인 공간상에 좌표 x, y, z를 갖는 시작점(seed point)이 어느 블록에 있는지, 그리고 그 블록의 어느 I, J, K index 셀에 위치해 있는지를 검색하는 단계, 해당 셀의 꼭지점으로부터 가중치를 곱하여 속도 벡터 $\vec{u}(\vec{\xi})$를 보간하는 단계, 그리고 일정한 시간 간격으로 적분을 하여 포인트의 위치를 구해내고 화면에 그려주는 단계를 거치게 된다.

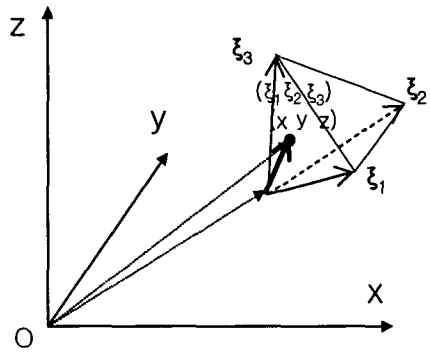


Fig. 8 Computational 공간의 좌표

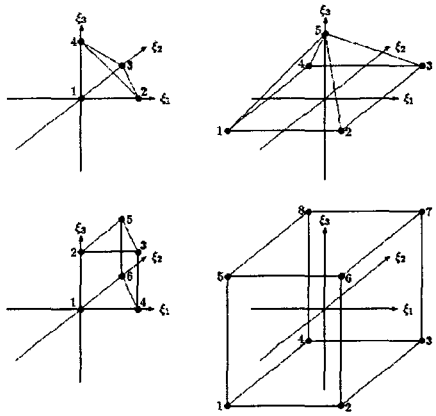


Fig. 9 Computational 공간의 standard 셀

먼저 검색과정에서 2차원은 해당 셀을 검색하기 위한 방법으로 area test[9]를 사용할 수 있지만 3차원에서는 이 방법이 적용되지 않는다. Strid 와 Rizzi[10] 등이 사용한 방법에 의하면 물리공간(physical space)의 임의의 점 $\vec{x}$에 해당하는 계산공간(computational space)의 좌표 $\vec{\xi}$ 를 구하기 위해 Fig. 8에서처럼 계산공간 좌표계를 설정하고 그 값을 구하여 해당셀에 대해서 Newton-Raphson 방법으로 수렴 여부를 조사하게 된다. 각 셀의 계산좌표계 (ξ_1, ξ_2, ξ_3)는 Fig. 9와 같이 그 셀이 속한 표준셀 타입에 따라 설정된다.

공간좌표와 계산좌표 간의 매핑은 식(1)과 같이 표현된다.

$$\vec{x}(\vec{\xi}) = \sum_j f_j(\vec{\xi})\,\vec{x}_j \tag{1}$$

여기서 $\vec{x}_j$ 는 j 번째 노드의 좌표 벡터이다. f_j 는 j 번째 노드에서는 1 이고 나머지에서는 0인 계수로서, 대표적인 tetra와 hexahedra 셀의 경우 식 (3), (4)와 같다.[1]

초기 $\vec{\xi}^{(0)}$는 추측한 셀의 중앙값을 취하고 식 (2)를 이용하여 반복 계산이 이루어진다.

$$\vec{\xi}^{(n+1)} = \vec{\xi}^{(n)} + \left(\frac{\partial \vec{x}}{\partial \vec{\xi}}\right)^{-1}(\vec{x} - \vec{x}^{(n)}) \tag{2}$$

▸ Tetrahedra

$$\begin{aligned} f_1 &= 1 - \xi_1 - \xi_2 - \xi_3 \\ f_2 &= \xi_1 \\ f_3 &= \xi_2 \\ f_4 &= \xi_3 \end{aligned} \tag{3}$$

▸ Hexahedra

$$\begin{aligned} f_1 &= \frac{1}{8}(1-\xi_1)(1-\xi_2)(1-\xi_3) \\ f_2 &= \frac{1}{8}(1+\xi_1)(1-\xi_2)(1-\xi_3) \\ f_3 &= \frac{1}{8}(1+\xi_1)(1+\xi_2)(1-\xi_3) \\ f_4 &= \frac{1}{8}(1-\xi_1)(1+\xi_2)(1-\xi_3) \\ f_5 &= \frac{1}{8}(1-\xi_1)(1-\xi_2)(1+\xi_3) \\ f_6 &= \frac{1}{8}(1+\xi_1)(1-\xi_2)(1+\xi_3) \\ f_7 &= \frac{1}{8}(1+\xi_1)(1+\xi_2)(1+\xi_3) \\ f_8 &= \frac{1}{8}(1-\xi_1)(1+\xi_2)(1+\xi_3) \end{aligned} \tag{4}$$

$\vec{x}$에 해당하는 셀에 대해서 반복을 하면 수렴하게 되고 수렴하지 않으면 이웃하는 셀을 다시 위의 절차를 반복하여 수렴하는 셀을 찾는다. 또 다른 방법으로 본 프로그램에 적용되고 있는 방법은 보다 빠른 검색을 위하여 먼저 min-max test[9]를 통해 가능성이 높은 셀 들을 찾아낸 다음 그 셀에 있는지를 조사하는 방법으로 I 방향에 대하여 각각 셀에 평행한 평면을 생각하면 normal vector와 추측을 하고 있는 $\vec{x}$ 와의 inner product를 계산한다.(Fig. 10)

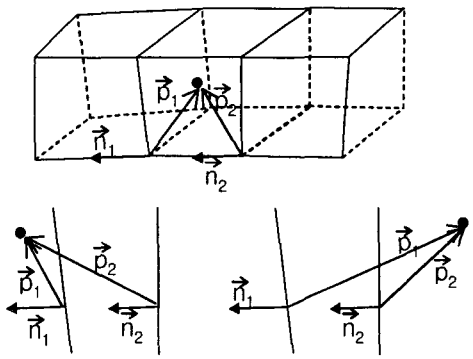


Fig. 10 셀 내부의 존재여부 판단

$\vec{x}$가 두 평면 사이에 있으면 이 둘을 곱했을 때 음의 값이 나오게 되고 밖에 있으면 양의 값이 나오게 된다. 이 과정을 나머지 J, K 방향에 대해서 거치고 만약 이들 중 양의 값이 나오면 바로 옆의 셀에 대해서 조사를 한다.

다음으로 찾아진 셀의 꼭지점 좌표와 $\vec{x}$로 속도 벡터 $\vec{u}(\vec{\xi})$를 보간하게 되는데, 2차원에서는 bilinear interpolation 방법을 사용할 수 있지만 3 차원에 적용하기에는 번거로운 작업이 많으므로 구해진 $\vec{\xi}$ 와 f_j로부터 식(5)와 같이 구한다.

$$\vec{u}(\vec{\xi}) = \sum_j f_j(\vec{\xi})\,\vec{u}_j \tag{5}$$

마지막으로 $\vec{u}$로부터 오일러 적분이나 런지쿠타 적분으로 $\vec{x}$를 구하여 전진을 하면서 자취를 저장하게 된다.

3.2.2 Stream ribbons

Stream ribbon은 streamline에 약간의 width를 주어 표현하는데 stream 방향의 vorticity를 나타낼 수 있다. 본 프로그램의 초기단계에서 시도했던 방법에 따르면 두 개의 streamline을 이어서 표현하고 있지만 변화가 심한 영역에서는 이들이 발산하는 경우가 있어 회전하는 정도를 표현하기에는 문제가 있음이 확인되었다. Visual3에서 사용하고 있는 방법은 하나의 streamline을 그려나가면서 stream 방향에 수직으로 일정한 길이의 타일을 붙여가면서 회전하는 정도를 다음 식(6)과 같이 구하여 적용한다.[1]

$$\frac{d\theta}{dt} = \frac{1}{2}(\vec{\omega} \cdot \vec{s})$$
$$\vec{s} = \frac{\vec{u}}{|\vec{u}|} \qquad (6)$$
$$\vec{\omega} = \nabla \times \vec{u}$$

3.3 탐침

탐침 방법에는 공간상의 어떤 점의 종속변수의 값에 대한 정보를 알 수 있도록 point probe 기능과 line에 대하여 관찰 할 수 있는 line probe, 그리고 streamline 상의 변화를 살펴 볼 수 있는 streamline probe 등이 있다.

4. DAVA 2.0 구성 및 적용 예

4.1 프로그램화면 구성

본 프로그램의 화면 구성은 Fig. 11과 같이 이루어져 있으며 여러 개의 영역으로 구성되어 있고, 각 영역의 기능은 다음과 같다.

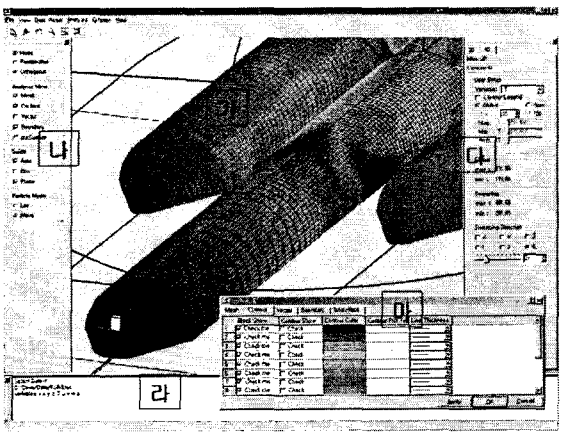


Fig. 11 프로그램 작업 환경의 예

가) Graphic viewport : 3차원 그래픽 처리 영역
나) Main control : 주요 plot 기능 on/off 및 데이터 컨트롤 기능
다) Analysis control and setup : 세부 데이터 분석 기능
라) State view : 데이터 정보, 작업 history
마) Multi-block detail setup : 다중블록 세부 설정

4.2 메뉴 체계

본 프로그램의 메뉴의 구성은 Table 1과 같이 되어 있는데 각 메뉴를 선택하면 세부 설정을 위한 다이얼로그창이 팝업된다.

Table 1 DAVA 2.0 메뉴

File	View	DataVisual	Analysis	Gridgen
New	Mode	Streamline	2D Analysis	1D
Open	Guide	Trace	3D Analysis	2D
Save Data	Active Control	Iso-Surface	Probe	3D
Save Screen				
Import				
Export				

4.3 Multi-block 설정 환경

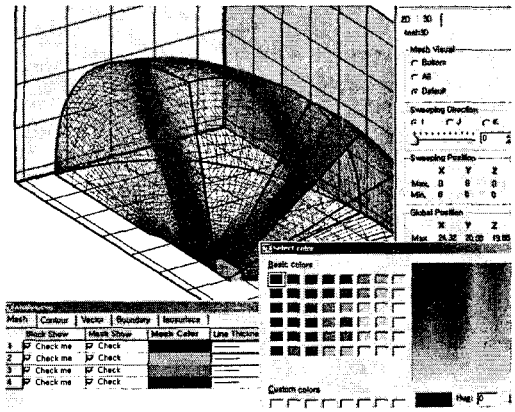


Fig. 12 Multi-block 환경 설정

Fig. 12은 4개의 블록으로 이루어진 다중블록 데이터의 Mesh plot 예이다. 다중블록 데이터는 데이터를 분석하는 데에 각각의 블록마다 Mesh, Contour, Vector, Boundary, Iso-surface 기능 등에 대하여 on/off 할 수 있어야만 효과적인 가시화 및 분석이 가능해 진다. 본 프로그램은 이를 고려하여 GUI 환경을 구성 하였으며 이러한 설정 값들이 MultiStructuredGrid 클래스에서 적용되어 각각의 개별 블록들에 적용되도록 하여 Mesh Color, Line Thickness, Index Skip 등의

기능을 사용자가 다양하게 설정 할 수 있도록 하고 있다. 아주 많은 설정 값들이 필요하여 프로그램이 복잡해질 수 있는데 객체지향 개념에 근거한 구조로 인하여 프로그램의 내부 구성이 견고하게 유지될 수 있었다.

4.4 Mesh Plot

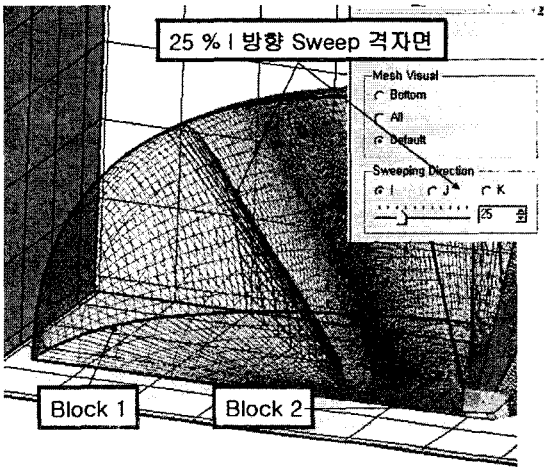


Fig. 13 Mesh sweep

대용량의 데이터를 효과적으로 가시화할 수 있도록 모든 격자점들을 망으로 이어 주는 모드와 물체 표면격자만 그려주는 모드, 그리고 각 블록의 외곽면 만을 그려주는 모드가 선택 가능하다. 데이터 크기를 텍스트와 그래픽 창에서 확인 할 수 있도록 되어 있고, 격자면의 index를 따라 격자를 움직이면서 격자의 형태를 관찰할 수 있도록 되어 있다. 이러한 기능들은 계산과정을 거친 데이터의 후처리 결과뿐만 아니라 격자 생성 후 만들어진 격자를 확인하는 데에도 유용하게 사용될 수 있다. Fig. 13에서 보는 바와 같이 어느 특정 블록의 격자계를 한 방향을 따라 차례로 격자면을 확인 할 수 있으며, 다른 기능과 함께 계산 결과와 격자와의 상관관계를 분석하는 데 도움이 될 수 있다.

4.5 Contour Plot

3차원 데이터에 대한 contour plot기능은 I,J,K 방향의 sweeping 기능과 공간절단면을 이용하는 X,Y,Z 방향 sweeping 기능으로 구현되었다. Fig. 14는 SSLV(Space Shuttle Launch Vehicle) 데이터를 공간절단면 contour plot 기능으로 가시화 한 예이다.

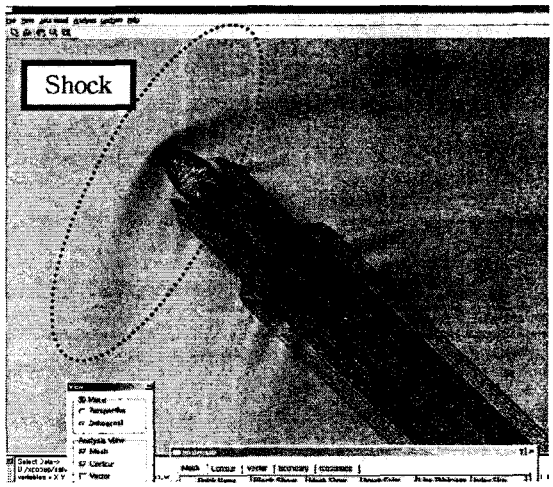


Fig. 14 Contour plot

본 프로그램(DAVA 2.0)은 최대값을 적색으로 최소값을 청색으로 표현을 하고 있는데 흑백의 명암으로 표현하여 실험에서 사용하는 슐리렌(Schlieren) 사진과 CFD 해석 결과와의 직접적인 결과 비교가 가능한 지에 대하여 추가 연구를 진행하고 있다.

4.6 Vector plot

Vector plot 또한 격자면 I,J,K Sweeping 기능과 X,Y,Z 공간절단면 sweeping 기능이 구현되었다. Vector plot에 사용하는 변수를 사용자가 선택하도록 되어 있고 벡터의 크기가 너무 작거나 클 경우를 대비하여 사용자가 벡터의 화면상 크기를 바꿀 수 있도록 하고 있다.

Fig. 15는 삼각날개(delta wing) 주변의 contour와 물체 주위의 속도벡터를 나타내고 있다. 유동변수와 X 방향의 공간 절단면에 속도벡터를 동시에 가시화함으로써 날개 주변 유동이 회전하고 있음을 확인할 수 있다.

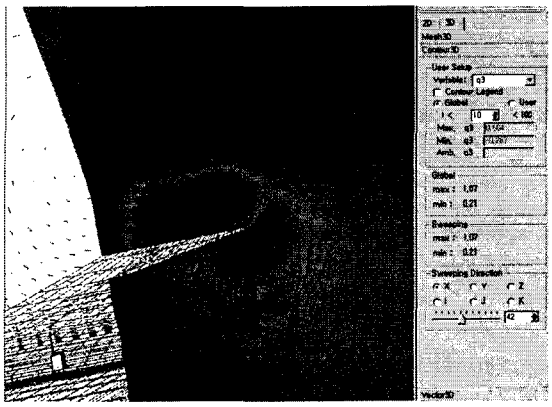


Fig. 15 Vector plot과 contour plot

4.7 Streamline plot

Streamline plot 기능이 구현된 예가 Fig. 16에 나와 있다. 3차원 공간에서 구현된 streamline인데 delta-wing 윗면 근처에서 leading-edge vortex를 보여주고 있다. 이러한 기능이외에도 CFD 계산 결과로부터 물리적인 의미를 찾을 수 있는 기능들이 추가 된다면 분석에 효과적인 도구로서 발전할 수 있을 것으로 여겨진다.

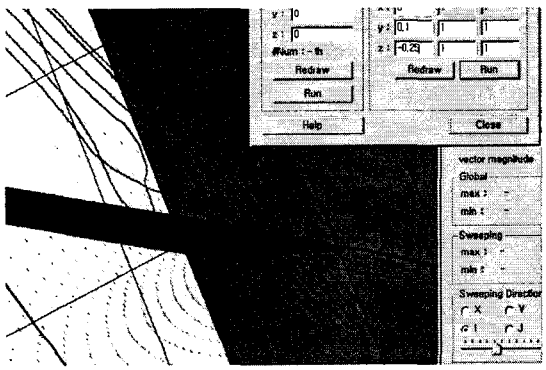


Fig. 16 Streamline plot

5. 결 론

본 논문에서는 유체 계산 결과 데이터의 효율적인 가시화 프로그램의 작성에 객체지향 개념의 도입이 필요한 이유를 살펴보았고, 현재까지 구현된 가시화 기능들과 구현 알고리즘 등에 대하여 설명하였다. 현재까지 구현된 본 프로그램의 기능이외에도 여러 가지 추가되어야 할 가시화 기능들이 있지만, 특히 CFD 분야에서 데이터 후처리시 많이 사용하는 분석기능을 추가함으로써 국내 CFD 및 관련 분야의 연구에 도움이 될 수 있는 효과적인 도구로 활용될 수 있기를 희망한다.

후 기

본 연구는 정보통신부 국가그리드기반 구축사업-그리드 시범 응용연구로 지원된 "그리드를 이용한 형상최적 설계기술 개발" 과제에 의해 수행된 연구 결과의 일부이며, 과제를 지원해 주신 관계자 여러분께 감사드립니다.

참고문헌

[1] R. Haimes and M. Giles. "Visual3 - A Software Environment for flow Visualization", 1991

[2] http://www.fluent.com

[3] http://www.kisti.re.kr

[4] http://www.Cactuscode.org

[5] 사종엽, 유종근, "3D 후처리 프로그램 개발에 관한 연구", 한국전산유체공학회 추계학술대회논문집, 1999

[6] 나정수, 김기영, 김병수 "입체구현기능을 지닌 데이터 분석 및 가시화 프로그램의 개발", 한국전산유체공학회 춘계학술논문집, 2002

[7] http://www.trolltech.com

[8] Edward Angel, Interactive Computer Graphics with OpenGL, ADDISON-WESLEY, p.500-514, 2000

[9] 옥영종, Data 후처리용 다기능 Software의 개발. 석사학위논문, 충남대학교, 2002

[10] T. Strid and A. Rizzi, "Development and use of some flow visualization algorithms", VKI Lecture Series on Computer Graphics and Flow Visualization in CFD, 1989