◎ 기술논문

객체지향형 2차원 유동 가시화 코드 개발

사 종 엽*1, 허 준 성*2

Code Development for Two-Dimensional Flow Visualization

Jong-Youb Sah, Jun-Sung Huh

The post-processor for two-dimensional flow visualization has been developed by using OOP(object-oriented programming) of Visual C++. User-friendly GUI(graphic user interface) has been built on the base of MFC(Microsoft Foundation Class). The number and order of variables can be specified by user because the input style is the free-format. The new variable can be defined and added to the variable list by using the various operators and functions.

Key Words: Flow Visualization(유동 가시화), CFD(전산유체역학), Post-Processor(후처리기), OOP(객체지향 프로그래밍), GUI(그래픽 사용자 인터페이스)

1. 서 론

복잡한 형상에 대한 유동 문제를 수치해석 방법으로 계산하고자 하는 경우, 문제의 형상 및 경계조건을 입력하는 전처리 장치, 해석 결과를 얻기 위한 계산 장치, 계산된 결과를 컴퓨터 그래픽으로 표현하여 주는 후처리 장치가 필요하게 된다. 계산 장치를 위한 수치해석 방법에 대한 연구는 대학 및 연구소에서 활발하게 진행되어 많은 양의 연구결과가 축적되어 있으나, 전처리기와 후처리기에 대한 연구결과는 상대적으로 미흡한 편에 속한다. 전처리기는 계산 장치에 필요한 계산 격자를 생성하기 위하여 복잡한 형상을 입력하여 필요한 블록으로 분할한 뒤 격자를 생성하는 장치로써, 계산 격자의 질이 계산 결과에 영향을 미치기 때문에 효율적인 격자를 확보하기 위하여 후처리 장치에 비하여 상대적으로 많은 연구가 진행되어 왔다. 이에 반하여 후처리 장치는 대학이나 연구소의 연구관심의 대상에서 다소 소외되어 있었으며, 주로 상용 소프트웨어의 후처리 장치를 빌어 쓰는 경우가 많았다. 그러나 계산된 결과를 더 유용하게 분석할 수 있는 도구에 대한 필요성이 커지면서, 후처리 장치에 대한 관심이 높아지고 있다.

본 연구에서는 교육용으로 사용 가능한 2차원 유동 가시화를 위한 후처리 장치를 개발하였다. 프로그래밍에 있어서 객체지향화 기법을 사용함으로써 복잡한 후처리 기능들을 모듈화하여 복합적인 기능들을 단순화하고, 유연한 확장성을 제공하였다. 객체지향화 기법은 변수와 함수를 결합한 클래스와, 클래스 간의 파생과 상속 등의 개념들을 제공하므로, 효율적이고 강력한 후처리 프로그램의 개발이 가능하다.

본 연구에서 개발된 프로그램 CAF2D/VisFlow V1.2는 유동 가시화에 있어서, 입력된 데이터를 이미지로 표현하는 단순 그래픽 기능뿐만 아니라, 데이터 속에 내포된 물리적 현상을 찾아낼 수 있도록 입력된 데이터를 가공 및 분석하는 확장 그래픽 기능들을 포함하고 있다. 그러므로, 공학 분야의 데이터 가시화에 다양하게 응용할 수 있다. 주요 특징으로는 사용자에 의한 데이터 포맷의 정의, 연산 및 함수를 이용한 새

* 2002년 10월 21일 접수

*1 정회원, 영남대학교 기계공학부
*2 정회원, (주)온디맨드소프트

Table. 1 Operators

operator	how to use
+	s = s + s v = v + v
-	s = s - s v = v - v
*	s = s * s v = s * v
/	s = s / s v = v / s

(s:scalar, v:vector)

로운 변수의 생성, 다중 블럭의 지원, particle trace, 다중 윈도우에 의한 데이터 간의 비교 등이 있다.

본 연구에서 개발된 2차원 유동 가시화 코드는 대학에서의 전산유체역학 교육을 지원하기 위한 목적으로 개발되었으며, 이러한 결과들을 통하여 후처리 장치 개발에 대한 국내 연구자들의 관심을 고조시켜 향후 경쟁력있는 후처리 장치가 국내에서도 개발될 수 있을 것으로 기대한다.

2. 데이터 처리

본 연구의 데이터는 입력, 가공, 출력의 순서로 처리된다.

2.1 데이터 정의 및 입력

컴퓨터를 사용한 수치해석의 결과나 실험에서 얻어진 데이터는 각자의 고유한 파일 포맷으로 저장되어 있다. 그러므로 이 데이터 파일을 후처리 프로그램에서 사용하고자 한다면 후처리 프로그램이 읽어들일 수 있는 데이터 양식으로 변환해 주는 새로운 프로그램이 추가적으로 필요하게 된다.

본 연구에서는 파일 포맷 변환을 위한 부가적인 작업을 제거하기 위하여 파일포맷방식을 제한하지 않는 free-format 방식을 채용하였다. 모든 데이터 파일의 포맷은 사용자가 원하는 대로 변수의 개수와 읽어들일 순서, 각 변수명을 정의할 수 있도록 하였다. 일반적으로 자주 사용하는 데이터 포맷들은 사용자들이 저장할 수 있도록 함으로써 사용자의 편리성을 증대시켰다.

데이터의 입력에서 격자 데이터와 수치계산 데이터를 분리시켜 입력할 수 있도록 하였다. 동일한 격자로 다양한 계산을 수행하는 예가 많이 있으므로 격자 데이터와 수치해석 결과 데이터를 분리시켜 다루는 것이 더욱 편리할 것이다.

Table. 2 Functions

function	how to use
SIN	s = SIN(s)
COS	s = COS(s)
TAN	s = TAN(s)
ASIN	s = ASIN(s)
ACOS	s = ACOS(s)
ATAN	s = ATAN(s)
SQRT	s = SQRT(s)
LOG	s = LOG(s)
LN	s = LN(s)
EXP	s = EXP(s)
ABS	s = ABS(s) s = ABS(v)
GRAD	v = GRAD(s)
DOT	s = (v) DOT (v)
DIV	s = DIV(v)
CURL	s = CURL(v)
NORM	v = NORM(v)
STREAM	s = STREAM(v)
VORT	s = VORT(v)

(s:scalar, v:vector)

데이터 파일에서 읽어들일 변수의 개수와 순서는 free-format 이므로 사용자들이 원하는 방식으로 결정하여 사용한다. 단, 변수명을 정함에 있어서 스칼라는 하나의 변수를 갖지만, 벡터는 각 성분별로 두 개의 변수를 갖는다. 벡터 변수는 변수명만 사용하면 벡터를, 벡터명에 [1]이나 [2]를 붙여 사용하면 벡터의 각 성분을 의미한다 (예; vel은 벡터 변수, vel[1]과 vel[2]는 벡터의 각 성분에 해당하는 스칼라 변수).

2.2 데이터 연산 및 가공

본 연구에서는 Table 1과 Table 2에서의 연산자와 함수를 이용하여 입력된 변수들로부터 새로운 변수를 생성할 수 있도록 하였다. 새로운 변수 생성에 사용하는 기존 변수명은 데이터 정의에서 입력하거나 내부적으로 설정된 것을 사용한다. Table 2의 함수 중에서, NORM은 단위벡터를, STREAM은 stream function을, VORT는 vorticity를 의미한다. 또한 본래 2차원 평면(x, y)의 벡터에 대한 CURL은 평면에 수직한 방향(z)의 벡터가 되지만, 2차원 평면 상에서 z 방향 성분은 벡터로 표현할 수 없으므로 스칼라로 표현하였다.

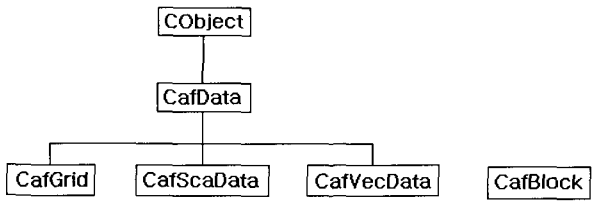


Fig. 1 Class hierarchy for data manipulation

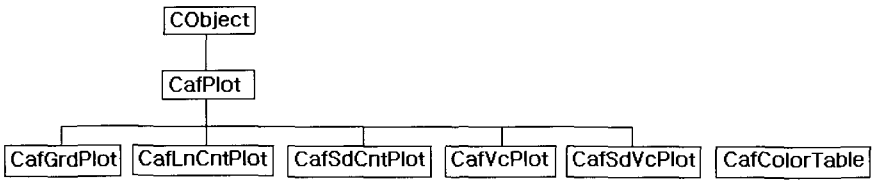


Fig. 2 Class hierarchy for graphic manipulation

원시변수를 사용한 2차원 유동 계산의 예를 들면, 속도벡터 v, 압력 p, 온도 T를 결과데이터에서 입력하여 각각의 변수에 대한 그래픽 결과를 출력한다. 만일 와도(vorticity)에 대한 그래픽 결과를 얻고 싶다면, 와도를 새로운 스칼라 변수(예를 들면 vort)로 지정하고 다음과 같이 함수 VORTICITY()를 이용하여 와도를 계산할 수 있다.

$$\text{vort} = \text{VORTCITY(v)} \tag{1}$$

$$\zeta = (\nabla \times \vec{v}) \cdot \vec{k} \tag{2}$$

만일 유함수(stream function)에 대한 후처리 결과를 보고자 한다면 유함수를 새로운 스칼라 변수(예를 들면 stre)로 지정하고 다음과 같이 함수 STREAM()을 이용하여 유함수를 계산할 수 있다.

$$\text{stre} = \text{STREAM(v)} \tag{3}$$

$$\nabla^2 \psi = -\zeta$$

$$= -((\nabla \times \vec{v}) \cdot \vec{k}) \tag{4}$$

만일 벽면에서의 열전달량을 알고자 한다면, 공기의 열전도 계수 k를 이용하여 전도에 의한 열전달량(예를 들면 cond)을 새로운 벡터 변수로 정의하고 연산자 *와 함수 GRAD()를 사용하여 열전달량을 계산한다. 벽면에서의 cond[1]은 x방향 열전달계수를 cond[2]는 y방향의 열전달계수를 얻는데 각각 사용할 수 있다.

$$\text{cond} = -k*\text{GRAD(T)} \tag{5}$$

$$\vec{q} = -k*\nabla T \tag{6}$$

2.3 데이터 가시화 및 출력

수치계산의 결과로써 입력된 변수와, 그리고 연산이나 함수에 의하여 추가적으로 정의된 변수들에 대하여 컴퓨터 그래픽을 이용한 가시화 결과를 출력한다. 스칼라 변수에 대하여 line 볼 수 있으며, shade contour를 이용하여 색변화에 의한 결과 출력을 볼 수도 있다. 전체적으로 빠른 판단을 필요로 하는 경우에는 후자를, 보다 자세한 분포가 필요한 경우에는 전자를 사용하는 것이 바람직하다.

벡터 변수의 경우, 각 성분을 이용하여 화살표 방향과 크기로 표현하는 arrow vector를 출력하거나, shade vector 출력을 이용하여 벡터 크기에 대한 정보를 색으로 표현할 수도 있다. shade vector는 색분포를 통하여 벡터 크기의 변화를 쉽게 예측할 수 있도록 도와 준다. 또한, 벡터가

속도인 경우로 가정하여 입자의 흐르는 경로를 예측하는 particle trace를 사용할 수 있다. 무게가 없는 질점에 대한 흐름 경로 계산은 4-stage Runge-Kutta 방법을 사용하였다.

$$x^{n+1} = x^n + \Delta x \quad (7)$$

$$y^{n+1} = y^n + \Delta y \quad (8)$$

여기서,

$$\Delta x = \frac{1}{6}(K_x^1 + K_x^2 + K_x^3 + K_x^4)$$

$$\Delta y = \frac{1}{6}(K_y^1 + K_y^2 + K_y^3 + K_y^4)$$

3. 프로그램 구성

3.1 개발환경

본 연구는 Microsoft Windows 95 이상의 OS를 채용한 IBM PC 호환기종을 대상으로 수행되었다. Linux OS를 채택한 컴퓨터에도 그래픽 사용자 인터페이스에 대한 약간의 수정만으로 사용이 가능하다. 객체지향 언어는 C++을 사용하였으며, 그래픽 사용자 인터페이스는 MFC(Microsoft Foundation Class) 라이브러리를 이용하였다. 컴퓨터 용량은 크게 제한받지 않으며, 펜티엄 II(120MHz) 64M RAM의 시스템에서도 무리없이 작동된다.

3.2 객체지향화

본 연구의 유동가시화 코드는 객체지향 기법을 사용하여 개발하였다. 특정 기능을 수행하는데 필요한 데이터와 함수들을 함께 묶어 캡슐화시킨 클래스를 사용함으로써, 복잡한 프로그램을 기능 별로 모듈화시키는 것이 가능하다. 이미 정의된 클래스로부터 새로운 확장 클래스를 파생시키는 것이 가능하며, 새로 파생된 클래스는 기본 클래스의 기능을 상속받고 새로운 기능들을 변경하거나 추가할 수 있다.

기본적으로 비슷한 기능을 수행하지만 조금씩 다른 기능들을 수행할 때, 기존 순차적 프로그래밍 기법은 조건절을 이용하여 경우의 수만큼 각 경우를 일일이 열거하여야 한다. 복잡한 프로그램에서는 가능한 모든 경우에 대하여 일일이 조건절에서 수행할 내용을 지정하여야 한다. 이러한 기존 방식의 순차적 프로그래밍은 방대하고 복잡한 구조를 갖게 되어 특정 기능 별로 모듈화하기가 어렵고, 새로운 기능을 변경하거나 추가할 때마다 유지관리면에서 유연성을 상실한다.

객체지향 프로그래밍 기법을 사용하면, 비슷한 기능들 중에 공통적인 개념을 추출하여 추상클래스를 만들고, 조금씩 다른 각 부분의 특성은 추상클래스를 파생시켜 각 특성을 변경하거나 추가함으로써, 모듈화가 용이하고 새로운 기능의 추가나 변경이 쉬워진다.

본 연구에서 중요한 두가지 클래스는 데이터 처리용 클래스와 그래픽 처리용 클래스이다. 데이터 처리용 클래스는 격자점의 좌표값이나 계산결과의 스칼라 및 벡터 변수들을 처리하는데 필요한 기능을 가지고 있으며, 데이터 처리에 있어서 공통적으로 필요한 특징만 추출하여 추상클래스 CafData 를 만들었다. 추상클래스는 데이터 클래스가 가져야 할 형식과 역할을 정의한 것으로 직접 데이터를 표현하는데 사용하지는 않는다. 격자 데이터를 표현하기 위하여 CafGrid 클래스를 추상클래스 CafData로부터 파생시켜 격자를 처리하는데 필요한 특정한 기능들을 추가한다. 압력이나 온도와 같은 스칼라 데이터를 처리하기 위하여 CafScaData 클래스를, 속도와 같은 벡터 데이터를 처리하기 위하여 CafVecData 클래스를, 각각 추상 클래스로부터 파생시킨다. Fig. 1은 데이터 처리용 클래스들 간의 파생관계를 잘 보여준다. 데이터의 입출력 기능을 예로들면, 추상클래스 CafData에 read()와 write()라는 기능의 형식을 지정하고, 실제 기능의 내용은 파생클래스 CafGrdData, CafScaData, CafVecData 별로 각각 자신에 적합한 read()와 write() 함수 내용을 지정한다. 클래스의 사용 시에는 스칼라 혹은 벡터 변수에 상관없이 read() 또는 write() 함수를 호출하면 각 상황에 적합한 기능들이 자동으로 호출된다. 다중 블록을 지원하기 위하여 블럭에 대한 정보와 기능을 CafBlock에 지정하고 이를 각 데이터 클래스와 연계하여 사용한다.

그래픽 처리를 위해서는, 추상 클래스 CafPlot에 그래픽 처리와 관련된 공통적인 특징만 추출하여 그래픽 처리용 클래스의 형식을 정의한 후, Fig. 2에서 보는 바와 같이 각 그래픽 기능에 따라 각각의 클래스를 파생시켜 특정 기능의 그래픽 내용을 추가한다.

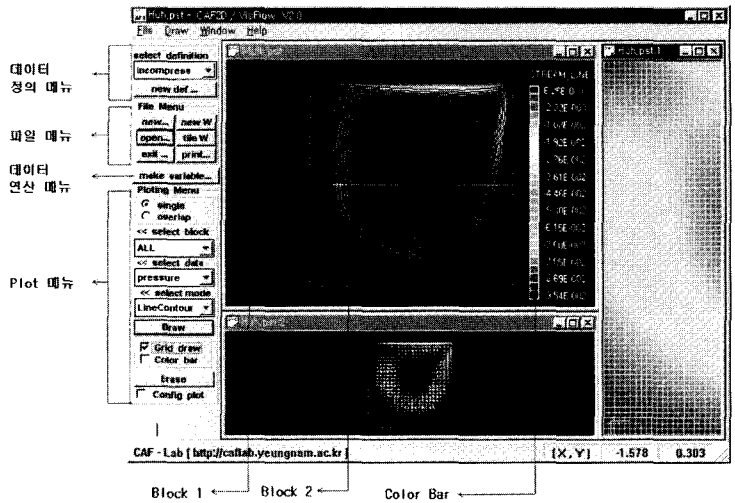


Fig. 3 Main window of graphic user interface

CafGrdPlot은 격자를 그리기 위하여 파생된 그래픽 처리 모듈이다. CafLnCntPlot는 line contour를 CafSd- CntPlot는 shade contour를 그리기 위한 그래픽 처리 모듈로써 스칼라 변수 혹은 벡터의 각 성분에 대하여 적용할 수 있다. 벡터 변수에 적용 가능한 CafVcPlot는 arrow vector를 CafSdVcPlot은 shade vector를 그리기 위한 그래픽 처리 모듈이다. 그래픽 처리의 한 예로 추상클래스 CafPlot에 draw()와 config()라는 기능의 형식을 지정하고, 실제 그래픽에서의 기능은 각 파생 클래스에서 각자의 역할에 적합한 draw()와 config() 함수 내용을 지정한다. draw()는 그래픽 출력을 config()는 그래픽과 관련된 파라미터 정보 설정을 담당한다. 클래스의 사용 시에는 그래픽의 종류에 상관없이 draw() 또는 config() 함수를 호출하면 각 그래픽 종류에 적합한 기능들이 자동으로 호출된다. 색테이블을 이용하여 색변화에 의한 shade 그래픽 출력을 다루기 위하여 별도로 CafColorTable 클래스를 설계하여 관련 그래프 모듈들과 연계되도록 하였다.

3.3 GUI 구성

본 연구를 통하여 개발된 프로그램의 운영 환경이 Fig. 3에 나타나 있다.

좌측 상단의 데이터 정의 메뉴는 변수 형식을 설정하는 것이다. 기존 정의된 양식을 사용하거나 새로운 양식을 정의하여 사용할 수 있다. Fig. 4는 2차원 유동해석 결과에 대한 변수 양식의 설정을 위한 리스트 박스의 예시를 보여준다. 새로운 양식이 필요하면 사용자가 추가할 수 있다. 데이터 개수와 순서가 결정되면 파일 메뉴에서 데이터 파일을 열거나 저장할 수 있으며, 필요한 그림을 인쇄할 수 있다. 새로운 창을 열 수도 있고 열린 창들의 배치를 조절할 수도 있다. 입력한 변수들로부터 연산과 함수를 사용하여 새로운 변수들을 얻고 싶다면 데이터 연산 메뉴를 선택하여 새로운 변수들을 쉽게 만들 수 있다. Fig. 5는 Table 1과 Table 2의 연산 및 함수를 이용하여 새로운 변수들을 정의하는 대화창을 보여준다.

Plot 메뉴에서는 현재 선택된 윈도우에 출력할 변수와 그래픽 종류를 선택한다. 출력할 변수의 개수는 단일(single) 혹은 복수(overlap) 중에서 선택할 수 있다. 단일변수(single)가 선택되면 기존 변수 대신에 새로운 변수 하나만이 지정되지만, 복수변수(overlap)가 선택되면 새로운 변수가 선택될 때마다 이전 선택된 변수에 새로운 변수들이 추가된다. 따라서, 하나의 그래픽 윈도우에 단일변수만 지정할 수도 복수의 변수를 지정할 수도 있다.

그래픽 윈도우는 반드시 하나의 CafGrid 클래스 객체의 포인터를 저장할 뿐만 아니라, 사용자가 지정한 변수들에 대하여 CafScaData 혹은 CafVecData 클래스의 객체들의 포인터를 이중링크 구조로 보관한다. CafScaData와 CafVecData

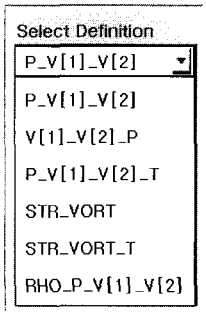


Fig. 4 Variable format

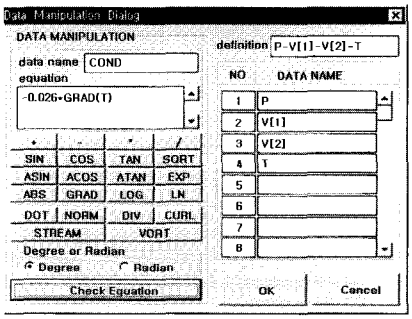


Fig. 5 Definition of new variable

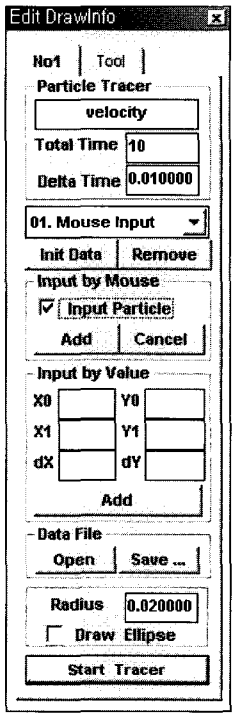


Fig. 6 Parameter for particle trace

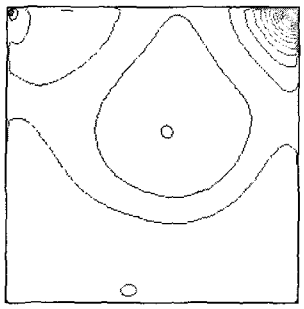
클래스들은 추상클래스 CafData로부터 파생되어 CafData의 속성을 상속받았으므로 포인터를 CafData 클래스형으로 사용할 수 있다. 단, 동일한 데이터 변수가 여러 그래픽 윈도우에 적용될 경우, 데이터 변수의 클래스 객체를 윈도우마다 만드는 것은 메모리 낭비를 초래하므로, 콘테이너 클래스를 이용하여 데이터 간의 이중링크를 구성한다. 콘테이너 클래스란 서로 간의 이중링크를 위한 인터페이스를 가지고 있고, 데이터 변수들의 클래스 객체들의 포인터를 보관하고 있는 클래스이다. 실제 데이터 변수들의 클래스 객체들은 주프로그램에서 이중링크에 의하여 관리함으로써 중복 생성되지 않도록 하고, 각 윈도우에서는 포인터에 대한 정보만을 콘테이너 클래스에 저장하여 사용한다.

그래픽 윈도우는, 각 윈도우에 선택된 변수(CafScaData와 CafVecData 클래스의 객체)들의 CafData의 포인터를 갖고 있는 콘테이너 클래스 각각에 대하여 CafPlot의 파생클래스들의 객체의 포인터들의 이중링크 구조를 보관하여야 한다. 예를 들어 어느 그래픽 윈도우에 하나의 벡터형 변수가 지정된 경우에, 하나의 벡터 변수에 대하여 두 개 이상의 그래픽을 구현할 수 있으므로, 벡터 변수의 객체 포인터를 담고 있는 콘테이너 클래스에 각 그래픽 클래스 객체들의 포인터를 이중링크 구조로 보관하여야 한다.

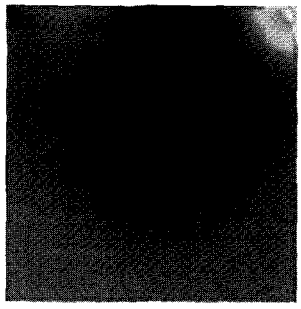
CafPlot에서 파생된 그래픽 클래스들은 각자 고유한 그래픽 파라미터 정보(config tool)를 가지고 있어서, 각 그래픽마다 필요한 정보를 조절할 수 있다. 예를 들어 하나의 스칼라 변수에 대하여 두 개의 line contour를 동시에 출력하는 경우, 두개의 CafLnCntPlot 클래스 객체들이 별도로 생성되어 각자의 고유한 파라미터 설정 데이터를 가지므로, 서로 다른 개수와 값의 등고선을 동시에 그리는 것이 가능하다. 격자 출력은 별도의 선택버튼(Grid draw)에 의하여 On/Off 할 수 있다.

3.4 Particle trace 파라미터 입력

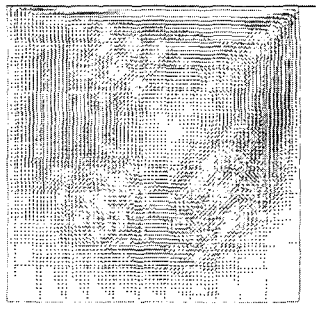
벡터 변수에 대한 particle trace는 벡터 변수가 속도 벡터라 가정하고 무질량 질점에 대한 경로선을 계산한다. Fig. 6은 경로선 계산을 위한



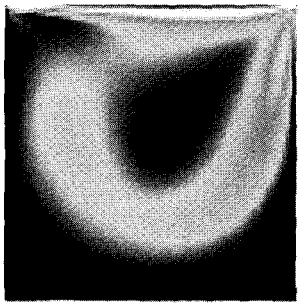
(a) line contour of scalar



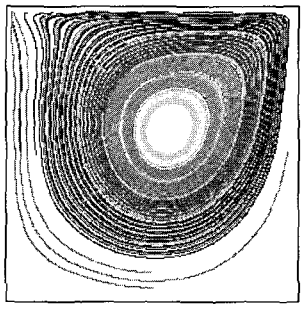
(b) shade contour of scalar



(c) arrow vector



(d) shade vector



(e) particle trace

Fig. 7 Graphics for variable

파라미터 설정 모듈을 보여준다. 이러한 경로선을 계산하기 위하여서는 질점의 초기 위치를 효율적으로 입력할 수 있어야 한다.

본 연구에서는 3가지 방법에 의하여 입자의 초기 위치를 결정한다. 첫째는, 마우스를 이용한 입력이다. 마우스 입력을 선택하고 마우스로 원하는 영역에 질점들을 설정한다. 원하는 부분에 질점 설정이 완료되면 이들을 하나의 그룹으로 선택한다. 마우스 입력은 정확한 위치 제어는 힘들지만, 대략적인 위치 제어에 유리하다. 둘째는 좌표값을 이용한 입력이다. 좌표값에 의한 질점 입력은 각 좌표 별로 시작과 끝, 그리고 증분을 입력하여 규칙적인 배열 형태의 질점들을 선택할 때 유리하다. 하나의 규칙적 배열 형태의 질점들을 하나의 그룹으로 선택한다. 마우스 입력과 좌표 입력은 각 그룹으로 분리되어 함께 사용할 수 있다. 그룹의 수는 원하는 만큼 분할하여 선택할 수 있다. 각 그룹 별로 입력된 질점들을 삭제할 수도 있다. 셋째는 파일 읽기에 의한 입력이다. 입력된 그룹들은 파일형태로 저장할 수 있으며, 저장된 파일로부터 읽어들여 불필요한 그룹을 삭제하거나 다시 필요한 그룹들을 마우스나 좌표입력으로 추가할 수 있다. 이상 세가지 질점 입력 방법은 함께 혼용하여 사용할 수 있다.

4. 결과 및 고찰

본 연구에서 개발된 가시화 프로그램에 의한 후처리 결과가 Fig. 7에 나타나 있다. line contour와 shade contour는 스칼라 변수나 혹은 벡터의 각 성분과 같이 하나의 스칼라 변수에 대하여 적용할 수 있다. arrow vector와 shade vector, particle trace는 벡터에 대하여 적용할 수 있다. 각 그래픽마다 각자 config tool을 가지며, 여기에서 그래픽과 관련된 파라미터들을 조정할 수 있다.

*line contour*는 등고선의 개수를 지정하고 각 등고선의 값을 수동이나 자동으로 설정할 수 있다. 설정된 등고선 파라미터는 별도로 파일에 저장하여 필요 시에 읽어들여 사용할 수도 있다. 이 것은 정확한 결과 비교가 필요할 때 사용한

다. shade contour는 color table에 지정된 색의 값으로 전체 영역을 색으로 덮는 것이다. 이러한 방식은 색을 보고 정확한 값을 예측할 수는 없으나 대략적인 분포를 예측하는데 편리하다. line contour에서도 color table을 이용하여 각 등고선의 값에 따라 등고선의 색깔을 바꾸어 준다.

arrow vector는 벡터형 변수를 화살표로 표시한 것이다. 화살표의 스케일은 수동으로 설정하거나 자동으로 설정할 수 있으며, 벡터 크기에 따라 color table을 이용하여 화살표의 색깔을 바꾸어 줄 수 있다. shade vector는 벡터 크기에 대한 shade contour를 출력한다. 이는 벡터의 크기의 대략적인 분포를 예측하는데 편리하다.

particle trace는 각 질점마다 color table을 이용하여 색을 바꾸어 출력할 수 있다. 모든 그래픽 처리에서 기본적으로 color table이 사용되며, color 테이블은 CafColorTable 클래스를 만들어 CafPlot으로부터 파생된 모든 그래픽 클래스에 연계되도록 설계하였다. overlap 기능을 이용하면 여러 그래픽들을 하나의 창 위에 중복 표현할 수 있다. 다중 블록을 지원하도록 설계되어, 각 블록 별로 다양한 그래픽 처리를 별도 지원한다. 다중 윈도우를 지원하므로, 여러 개의 그래픽 윈도우를 열고 다양한 그래픽 작업을 동시에 수행하고 비교하는 것이 가능하다.

5. 결 론

본 연구를 통하여 개발된 유동가시화 코드는 객체지향화 기법을 사용하였다. 강력하고 편리한 후처리 장치를 개발함에 있어서, 순차적 프로그래밍 기법을 사용하는 복잡한 프로그램 구조와 달리 모듈화를 통한 단순한 구조를 가질 수 있으므로, 변경이나 확장에 유연한 후처리 장치를 개발하는 것이 가능하다.

2차원 유동 가시화 코드 CAF2D / VisFlow V1.2는 국내 대학에서 전산유체역학 교육을 위한 후처리 장치로 개발되었다. 향후 후처리 장치에 대한 국내 연구자들의 관심을 고취시켜 국내에서 경쟁력있는 후처리 장치가 개발될 수 있도록, 후처리 장치에 대한 다양한 실험적 기법들이 채용되어 테스트되었다.

전통적인 기계공학분야에 있어서 본 연구와 같은 응용 프로그램의 개발은 정보처리 분야에 과도하게 집중된 IT 연구 역량을 분산시켜 균형적이고 경쟁력있는 IT 발전에 기여할 수 있을 것이다.

본 연구를 3차원 유동 가시화로 확장하는 것은 데이터 구조 상의 차이로 인하여 2차원을 3차원으로 단순확장하는데 어려움이 따른다. 3차원 유동 가시화를 위하여서는 별도의 데이터 구조를 설계하여야 한다. 3차원 가시화에 2차원 가시화를 포함시키면 2차원 가시화만큼의 GUI 구축에서의 효율성을 얻기 어렵다. 따라서, 3차원 유동 가시화는 별도의 연구를 통하여 분리되어 수행되었으며, 현재 개발이 완료 단계에 있다.

본 연구는 대학과 연구소에서의 교육 및 훈련, 연구를 지원하기 위하여 개발되었으므로, 상업적 목적이 아닌 경우에는 프로그램을 자유로이 다운로드 받아 사용할 것이다. 인터넷 상에서 CAF2D / VisFlow V1.2를 다운로드할 수 있는 사이트는 다음과 같다.

1. 한국전산유체공학회 : http://www.kscfe.or.kr
2. 영남대 컴퓨터응용유체연구실 : http://caflab.yu.ac.kr

후 기

본 연구는 한국학술진흥재단 연구비 지원에 의하여 수행되었습니다.

참고문헌

[1] 김용성, Visual C++ 6 완벽가이드, 영진 (2002).
[2] 사종엽, 유종근,"3차원 후처리 프로그램 개발에 관한 연구," 1999년도 한국전산유체공학회 추계학술대회 논문집, (1999), pp.163-166.
[3] 신영수, 김현식, 3차원 그래픽 - C언어로 배우는 이론과 알고리즘, 가남사 (1991).
[4] John Vince, 3-D Computer Anima- tion, Addison-Wesley (1992).
[5] Hans J. Lugt, Vortex Flow in Nature and Technology, John Wiley & Sons (1983).
[6] Martti Mantyla, An Introduction to Solid Modeling, Computer Science Press (1988).
[7] Paul S. Heckbert, Graphics GEMS IV, AP Professional (1994).