韓國海洋工學會誌 第13卷 第1號, pp.11~17, 1999. 2

◉ 論 文

3차원 정렬 유한요소 생성 코드 개발에 대한 연구

김 진 환*

(98년 10월 16일 접수)

A Study on the Development of a Three Dimensional Structured Finite Elements Generation Code

Jin Whan Kim*

Key Words : Finite Element(유한요소), Serendipity Basis Function(시렌디피티 기저함수), Hierarchical Shape Function(계층 형상함수), Structured Grid(정렬격자), Progressive Ratio(점진적 비율), Vertex(꼭지점), Edge(변), Degree(차수)

Abstract

A three dimensional finite element generation code has been developed attaching simple blocks. Block can be either a quadrature or a cube depending on the dimension of a subject considered. Finite element serendipity basis functions are employed to map elements between the computational domain and the physical domain. Elements can be generated with user defined progressive ratio for each block. For blocks to be connected properly, a block should have a consistent numbering scheme for vertices, side nodes, edges and surfaces. In addition the edge information such as the number of elements and the progressive ratio for each direction should also be checked for interfaces to have unique node numbers. Having done so, user can add blocks with little worry about the orientation of blocks. Since the present code has been written by a Visual Basic language, it can be developed easily for a user interactive manner under a Windows environment.

1. 서 론

일찍이 Barfield[1]는 격자간의 직교성의 중요성을 논한바 있으며, 등각 사상법을 이용하여 경계에서 격자선이 직교되게하는 수치적 기법을 제안한 바 있으나, 이는 이차원적 가하학적 형상에 한정되었다. 이후 J.F. Thompson[2]을 비롯한 많은 연구자들의 격자 생성법과 응용 사례들을 통하여 이의 중요성은 널리 알려져 있으며, 현재에도 CFD 혹은 CAE 분야에 종사하는 연구자들은 자

* 종신회원, 동의대학교 기계공학과

체의 격자 생성 코드를 개발하였거나 또 개발 중에 있다[3,4,5]. 유한요소 계산을 위하여는 요소 생성이 선행되어야 하며, 본 연구는 때늦은 감이 있으나 이를 위하여 먼저 3차원 육면체 정렬 요소 생성에 대한 기법을 다루고자 한다. 요소를 생성하기 위하여는 먼저 요소가 무리없이 생성될 수 있는 간단한 블록을 정의하고, 블록과 블록간의 위상학적 및 기하학적 관계가 정의되어야 한다. 일단 블록이 정의되면 그 블록내에서 요소는 보간 다항식을 이용하여 요소의 절점들이 결정될 수 있으며, 본 연구에서는 Serendipity 형상함수를 사용하여 보간하였다. Serendipity 형상함수를 이용하면 블록의 꼭지점과 변상의 점들만으로서 블록 내부와 표면에 대한 절점들의 좌표를 결정할 수 있으므로, 블록 구성에 필요한 좌표점들의 입력을 최소화할 수 있다. 또한 유한요소법의 특징은 형상함수를 통하여 고차의 수치해를 사용자의 별다른 노력없이 계산해 낼 수 있는 것인데, 이를 위하여는 일반적으로 고차 다항식에 상응하는 절점들을 요소가 가지고 있어야 한다. 이렇게 되면 너무나 많은 절점들이 필요하게 되어 최적의 요소 생성에 대한 이점이 없어지게 되어, 본 연구에서는 Devloo 외 2인[6]이 제시한 바와같이 계층(Hierarchical) 형상함수 혹은 Chebyshev 형상함수를 사용할 것을 전제로하여 요소의 기하적 형태를 2차 Lagrangian 사변형 혹은 육면체 요소로 한정하였다. 2차 Lagrangian 요소란 기하학적으로 이차원일 경우 9개의 절점을 필요로 하고 3차원일 경우 27개의 절점을 필요로 한다. 이러한 요소에 Hierarchical 혹은 Chebyshev 형상함수를 사용하면 절점의 증가없이 고차의 수치해를 계산할 수 있으므로 이 점은 추후 유한요소 계산법의 개발에 아주 유용하리라 사려된다.

최근 개인용 컴퓨터의 성능 향상으로 그래픽 환경의 이용이 상당히 용이해지고 있으며, 이에 따라 자료(Data)의 입력 및 수정도 사용자와 상호 대화형식(Interactive manner)으로 변하고 있는 추세이다. 이런 환경에서 코드 개발을 위한 언어들이 여러 가지가 있으며 본 연구에서는 Visual Basic 언어를 선택하였으며, 그 이유는 컴퓨터 비 전문가인 일반 공학도들이 쉽게 접근할 수 있기 때문이며 또한 객체 지향적(Object oriented) 코드 개발도 용이하리라고 보기 때문이다[10].

2. Block의 구성

3차원 육면체 블록은 8개의 꼭지점(Vertex), 12개의 변(Edge) 그리고 6개의 면(Face)으로 구성되어 있다. 만일 블록이 곡면체이면 사용자는 2차 보간을 할 것인지 혹은 3차 보간이 필요할 것인지에 대하여 결정을 해야 한다. 보간 차수에 따른 입력 절점의 수는 Serendipity 형상함수를 사용할 경우 변의 수와 동일하며 각 변에 절점이 추가된다. 따라서 2차 보간의 경우 20개의 절점, 3차 보간의 경우 32개의 절점이 요구되며, 이는 각 면과 내부의 절점이 필요하지 않은 관계로 최소 가능한 절점 입력 수이리라 사려된다. 본 연구에서 사용한 절점 번호 기입 순서는 Fig.1에, 변의 순서와 방향성(orientation)은 Fig. 2에, 면의 순서와 방향성은 Table 1에 각각 나타내었다. 블록을 정의하기위하여 사용된 Visual Basic 언어의 Type 변수는 다음과 같다.

```
Type BlockType
Type As String: Npoints As Integer: GeomDeg As Integer: ElemDeg(1 to 2) As Integer: Points(1 to 32) As Integer: Edges(1 to 12) As Integer: Faces(1 to 6) As Integer: Nelem(1 to 3) As Integer: ProgR(1 to 3) As Single: Boundary(1 to 6) As Integer: Variable(1 to 10) as integer:
End Type
```

상기한 변수들 중 "GeomDeg"는 블록 형상을 정의하기 위한 보간 차수를, "ElemDeg()"는 요소의 형상 차수와 해의 차수를, "Points()"는 블록을 정의하는 절점 번호를, "Edges()"는 변 번호를, "Faces()"는 면 번호를, "Nelem()"은 블록의 각 방향에 따른 요소의 수를, "ProgR()"은 블록의 각 방향에 따른 요소의 점진적 비율(progressive ratio)을, "Boundary()"는 경계조건의 형태를, "Variable()"은 준식(Governing equation)을 구성하는 변수를 각각 의미한다. 여기서 블록의 각 방향은 국부 좌표계(local coordinate system)의 방향 즉, 블록을 정의하는 절점 번호 부여에 따른 방향을 의미한다. 따라서 서로 인접한 블록이라도 각각 정의된 방향은

일반적으로 다르며, 인접 블록간의 상호 연결 관계(connectivity 혹은 topology)가 잘 파악되어야 한다. 본 연구에서는 면의 연결관계보다 변의 연결관계가 더 기본적이라고 사려되어 변에 대한 정의를 다음과 같이 하였다.

```
Type EdgeType
Degree As Integer: Points(1 to 4) as
Integer: Nelem As Integer: ProgR As Integer
End Type
```

변의 정의에 사용된 변수들의 의미는 블록의 정의와 동일하다. 제 1번 블록이 정의되면 이에따라 12개의 변이 정의된다. 제 2번 블록부터는 서로 공유하는 변이 있는지를 조사하여 만일 있으면 그 방향성(orientation)과 점진적 비율(progressive ratio)이 서로 같은 지의 여부를 검토하게 한다. 공유변의 방향성이 서로 다르면 요소를 분할한 후 순서를 역으로 함으로 해서 인접 블록간의 연결성을 갖게 하였다.

참고로 Fig. 1에 정의된 절점 순서에 따른 2차 serendipity 함수는 다음과 같다.

$$\psi_i = 0.125\ (1 + \eta_1\eta_1(i))\ x\ (1 + \eta_2\eta_2(i))(1 + \eta_3\eta_3(i))\ x\ (\eta_1\eta_1(i) + \eta_2\eta_2(i) + \eta_3\eta_3(i)),\quad i=1,2,3,4,5,6,7,8$$

$$\psi_i = 0.25\ (1 - \eta_1^2)(1 + \eta_2\eta_2(i))\ x\ (1 + \eta_3\eta_3(i)),\quad i=9,11,13,15$$

$$\psi_i = 0.25\ (1 - \eta_2^2)(1 + \eta_1\eta_1(i))\ x\ (1 + \eta_3\eta_3(i)),\quad i=10,12,14,16$$

$$\psi_i = 0.25\ (1 - \eta_3^2)(1 + \eta_1\eta_1(i))\ x\ (1 + \eta_2\eta_2(i)),\quad i=17,18,19,20$$

여기서 i는 블록을 정의하는 절점 번호이며 $\eta_1(i)$, $\eta_2(i)$ 및 $\eta_3(i)$는 i 절점의 각 방향의 값을 의미한다. 3차 serendipity 함수는 아래와 같이 정의된다.

$$\psi_i = (1/64)\ (1 + \eta_1\eta_1(i))\ x\ (1 + \eta_2\eta_2(i))(1 + \eta_3\eta_3(i))\ x\ (9(\eta_1^2 + \eta_2^2 + \eta_3^2) - 19),\quad i=1,2,3,4,5,6,7,8$$

$$\psi_i = (9/64)\ (1 - \eta_1^2)(1 + 9\eta_1\eta_1(i))\ x\ (1 + \eta_2\eta_2(i))(1 + \eta_3\eta_3(i)),\quad i=9,10,13,14,17,18,21,22$$

$$\psi_i = (9/64)\ (1 - \eta_2^2)(1 + 9\eta_2\eta_2(i))\ x\ (1 + \eta_1\eta_1(i))(1 + \eta_3\eta_3(i)),\quad i=11,12,15,16,19,20,23,24$$

$$\psi_i = (9/64)\ (1 - \eta_3^2)(1 + 9\eta_3\eta_3(i))\ x\ (1 + \eta_1\eta_1(i))(1 + \eta_2\eta_2(i)),\quad i=25,26,27,28,29,30,31,32$$

Table 1. Face definition for a block

Face # (name)	Edge numbers	Vertices
1 (Bottom)	1, 2, 3, 4	1, 2, 3, 4
2 (Top)	5, 6, 7, 8	5, 6, 7, 8
3 (Front)	1, 10, -5, -9	1, 2, 6, 5
4 (Right)	2, 11, -6, -10	2, 3, 7, 6
5 (Back)	3, 12, -7, -11	3, 4, 8, 7
6 (Left)	4, 9, -8, -12	4, 1, 5, 8

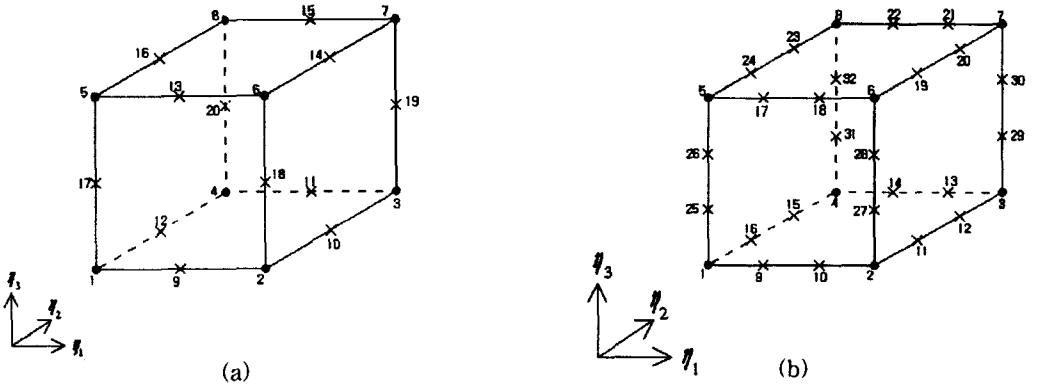

Fig. 1 A node numbering scheme defining a Block. (a) Block of quadratic degree (b) Block of cubic degree.

3. 요소의 구성

계산 영역에서 각 방향으로의 요소의 수와 점진적 비율에 따라 블록 내부에 요소를 생성한 후, 요소의 절점들은 Serendipity 형상함수를 통하여 실제 영역으로 다음과 같이 사상된다.

$$x_{ij} = \sum \psi_p(\eta_i, \eta_j)\, X_p$$
$$y_{ij} = \sum \psi_p(\eta_i, \eta_j)\, Y_p \qquad (1)$$

여기서 x_{ij} 및 y_{ij}는 (i,j) 격자점의 x 및 y 좌표, X_p 및 Y_p는 블록의 p번째 절점의 x 및 y 좌표, 그리고 $\psi_p(\eta_i, \eta_j)$는 (η_i, η_j)에서 p번째 정의된 serendipity 형상함수의 값이다. Visual Basic code에서는 요소를 다음과 같이 정의하였다.

```
Type ElementType
Type As String: NnodL As Integer: NdofL As Integer: GeomDeg As Integer: SolnDeg
As Integer: Nodes(1 to 20) As Integer: Dofs(1 to 100) As Integer: Boundary(1 to 6) As Integer: Variable(1 to 10) As Integer: End Type
```

여기서 "NnodL"은 요소를 정의하는 절점수, "NdofL"은 요소에 부여된 자유도, "GeomDeg"와 "SolnDeg"는 요소의 기하학적 및 수치해의 차수, "Nodes()"는 요소에 부여된 절점 번호, "Dofs()"는 요소에 부여된 자유도 번호, 그리고 "Boundary()"는 요소의 경계조건, "Variable()"은 변수정의를 의미한다. 요소의 절점 번호 순서는 혼동을 피하기 위하여 블록 정의의 경우와 동일하게 하였다. 임의의 블록에서 생성된 절점들은 다시 인접 블록과의 경계면에서의 공유점들에 대하여 조사하여 동일 절점 번호를 부여해야 한다. 그런데 이 작업은 예상외로 코드 구성에 어려운 점이 많아 현재에는 절점의 좌표에 대하여 허용공차를 주어 공차 이내의 절점에 대하여는 동일 번호를 부여하기로 하였다. 실제로 너무 작은 요소는 계산상 오차를 일으키기 쉽기 때문에 사전에 이를 방지하는 것이 타당하다.

현재 사용하고 있는 점진적 비율(ProgR)에 대한 개념은 다음과 같다. 만일 현재의 요소 길이가 Δx라면 다음 요소의 길이는 ProgR * Δx이 된다. 이를 수행하기 위하여 주어진 길이에서 최소 Δx를 아래와 같이 구한다.

sum = Σ (i=1 to N) ProgR^(i-1) ,

N = the number of elements

최소 Δx = (전체길이) / sum

그러면 전체길이는 아래와 같이 ProgR에 대한 등비급수가 된다. 즉

전체길이 = (최소 Δx) * (1 + ProgR + ProgR^2 + ... + ProgR^(N-1))

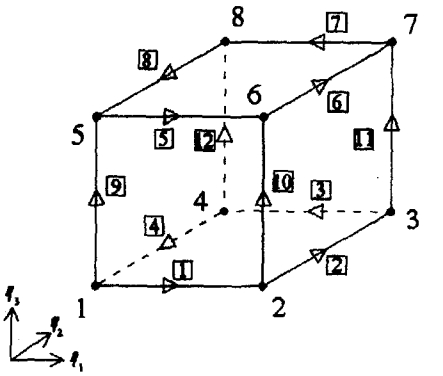

Fig. 2 An Edge numbering(numbers in box) scheme and Orientation(shown as arrows) for a block.

상기한 점진적 비율은 각 블록의 계산영역에서 각 방향의 요소 수에 따라 적용되며, 여기서 생성된 요소는 식(1)에 의한 사상을 거쳐 실제 좌표값을 가지게 된다. 블록 표면에 생성된 절점들에 대하여는 공유점들이 있는지의 여부를 조사하여 절점 번호를 재조정한 후 다음 블록으로 진행된다.

4. 실제 예를 통한 검증

현재 개발된 코드를 검증하기 위하여 아래 2개의 간단한 예제를 통하여 살펴 보기로 한다. 첫째는 선형 블록들의 조합으로서, 3개의 육면체와 1개의 4면체 블록들이 요소 수와 점진적 비율에 따라 인접 블록간의 상호 연결 상태를 확인하는 것이 주 목적이다. 두 번째는 심한 곡면에 대한 serendipity 함수를 통한 블록 분할이 어떠한지에 대한 것이 관심의 대상이 되며, 원심펌프 회전차의 유체 통로에 대한 자료[7]를 이용하였다. Table 2에는 선형 블록들에 대한

주요 입력자료를 나타내었으며 각 블록에 대한 절점 번호와 각 절점에 대한 좌표값들은 생략하였다.

4.1 선형 블록에 대한 검토

Fig. 3(a)에 각 블록들을 나타내었으며, Fig. 3(b)에는 생성된 요소들을 보여주고 있다. 제1번 블록의 제3 방향의 요소 수는 4이며, 제2번 및 3번의 동일 방향의 요소 수는 3이다. 이는 사용자의 입력 잘못으로 간주되어 코드는 자동으로 이를 무시하고 먼저 정의되는 블록의 정보에 우선권을 주도록 되어 있다. 이와 관련된 점진적 비율도 먼저 정의되는 정보가 우선한다. 제4번 블록은 평판 요소 모델로서 제1 방향의 요소 수는 2이지만 이미 제1번 블록에서 동일 방향으로 요소 수가 4개로 지정되어 제3번 블록에 정보가 전달되어 다시 제4번 블록까지 그 정보가 전달됨을 보여주고 있다. 따라서 선형 블록들에 대하여 현재의 코드는 원활한 수행을 하고 있음을 알 수 있다. 현재 코드에서 사용한 점진적 비율에 대하여 살피면, 제1번 블록의 제1방향은 1.2, 제2방향은 0.8로서 대체로 요소 길이의 변화는 완만함을 보이고 있으나, 제4번 블록의 제2방향은 0.5로서 그 방향으로 요소 길이가 급격히 줄어듬을 볼 수 있다. 요소의 면적이 인접 요소의 면적과 상당한 차이가 있으면 행렬 계산에 있어서 오차 발생을 크게할 위험이 있기 때문에 이 비율은 1.2와 0.8 사이에서 사용함이 적당할 것 같다.

4.2 원심펌프 회전차의 유체통로에 대한 검토

Table 2. Input Data Examples for Linear Blocks.

Block #	1	2	3	4
Block Type	Cube	Cube	Cube	Quad
Geom. Degree	1	1	1	1
Number of Points	8	8	8	4
Elem. Geom.Deg	1	1	1	1
Elem. Soln. Deg	1	1	1	1
# of elem. in 1 dir.	4	4	2	2
# of elem. in 2 dir.	6	3	5	5
# of elem. in 3 dir.	4	3	3	1
Prog. ratio in 1 dir.	1.2	1.0	1.0	1.5
Prog. ratio in 2 dir.	0.8	0.5	1.0	0.5
Prog. ratio in 3 dir.	1.0	1.8	1.0	1.0

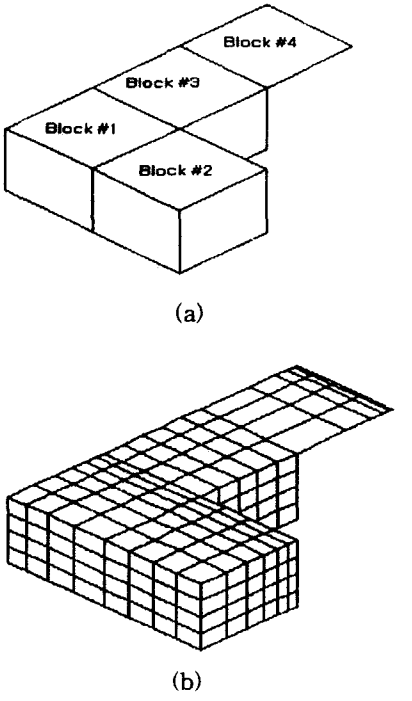

Fig. 3 A plot of linear block example
(a) Blocks (b) Elements

Fig. 4에는 유량 1.0 m^3/min, 양정 20.1 m 그리고 회전수 1800 rpm에 대한 1단 원심펌프에 대한 회전차 설계 자료를 AutoCad에서 그리게 한 것으로서, 깃이 반경방향으로 상당한 회전형태를 가지고 있음을 보여준다.

Fig. 5에는 2개의 블록을 깃의 반경 방향으로 나누고 각 블록의 보간 차수가 2로서 깃사이의 통로에 대한 것이다. 여기서 굵은 선으로 그린 면은 각 블록의 윗 부분을 나타내고 가는 선으로 그린 면은 아래 부분을 나타낸다. 유체의 입구측은 ABA'B'으로 정의되는 면이고 출구측은 CDC'D'으로 정의되는 면이다. 그림에서 보듯이 두 블록은 경계에서 대체로 잘 접하고 있음을 보이고 있다. Serendipity 함수는 C^0 함수가족이므로 곡선 양 끝단에서 부드럽게 연결됨을 보장하지 않으며, 부드러운 연결이 엄격히 요구된다면 블록 보간 함수를 C^1 함수가족 중에서 찾아야 하며 이는 추후 보완해야 할 사항이다. 현 예제의 검토에서 Serendipity 함수의 일반적

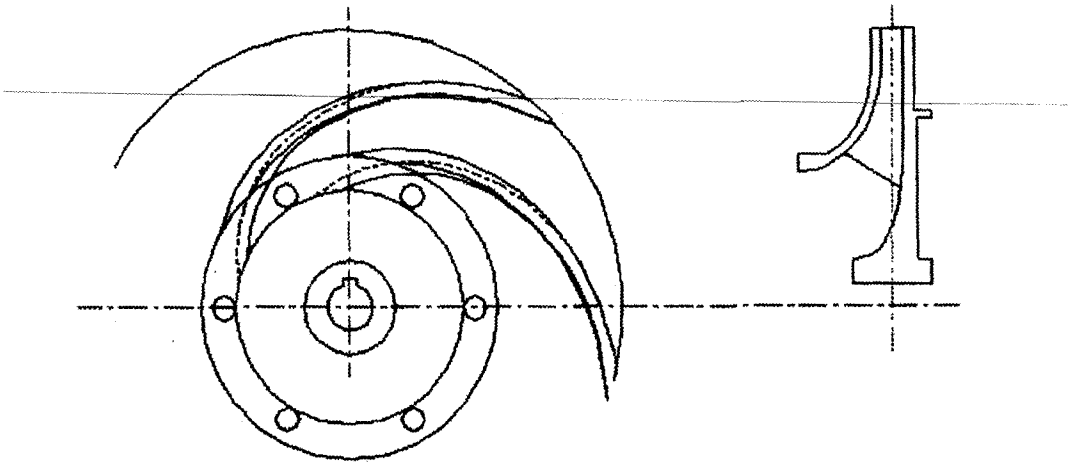

Fig. 4 A Cad Drawing of a centrifugal pump impeller.

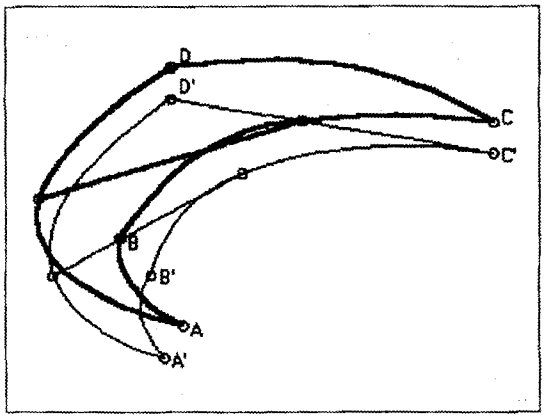

Fig. 5 A top and bottom view of a flow passage in a centrifugal pump impeller (Top plane drawn by a thick pen, bottom plane drawn by a thin pen).

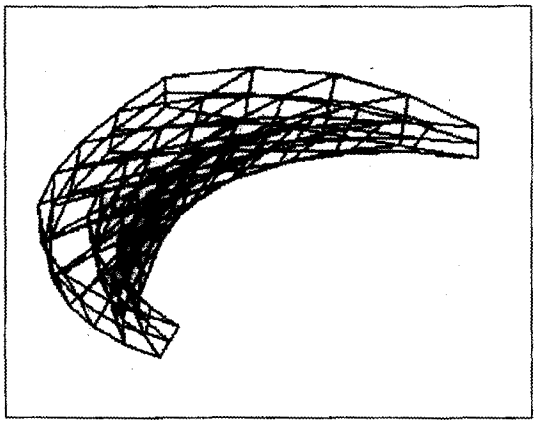

Fig. 6 Elements generated for a flow passage in a centrifugal pump impeller(4x4x2 elements in each block).

사용에 무리가 없음을 알 수 있다. 원심펌프 회전차와 같은 극심한 곡면에 대하여 아직 은면(hidden surface) 처리가 미흡하나, 이는 문제의 본질은 아니며 추후 OpenGL 혹은 적당한 방법을 통하여 해결할 예정이다.

5. 결 론

본 연구는 기본 블록 단위에서 생성된 요소들을 조합함으로서 다양한 형상에 쉽게 적용될 수 있는 3차원 유한 요소 생성 코드 개발에 대한 것으로, 현재 코드는 계산영역에서 실제 영역으로의 사상을 위하여는 serendipity 함수 그리고 사용될 요소는 사변형 혹은 육면체로 한정하였다. 이는 일반

사용에는 충분하리라 본다. 선형 블록과 곡면 블록에 대한 검토 결과, 본 코드는 최소의 입력자료로서 요구되는 요소를 생성할 수 있음을 알았다. 부드러운 곡면을 위한 보간함수의 발굴, 다양한 형상의 블록을 통한 요소의 생성 및 일반 다면체 요소의 생성등은 모두 어려운 과제들로서, 추후 하나 하나씩 연구해 나가야 할 것으로 사려된다. 현재 코드는 Visual Basic으로 작성되어 Windows 환경하에서 사용자와의 대화형식으로 발전되기에 용이하며, 또 다양한 형태의 요소들을 각각 객체(Object)로 간주한다면 객체지향적 코드 개발에도 적합하다.

최근에는 비정렬 격자 생성을 통한 3차원 해석이 시도되고 있으나, 고차차분법에는 적용이 쉽지 않으며 해의 수렴성도 정렬 격자계에 비하여 떨어진다고 보고되고 있다[11]. 더욱이 정렬격자계의 장점은 해의 풀이과정의 유연성에 있으며, 고전적인 부구조화(substructuring) 기법 그리고 최근의 영역분할법(domain decomposition method) 및 병렬계산법들은 특히 블록 단위로 구성된 격자계에 적합하다. 따라서 현재의 단순 블록을 통한 정렬 유한요소 생성 연구는 계산 기법과 관련하여 중요한 의미가 있다고 하겠다.

참고 문헌

1) W. D. Barfield, "Numerical method for Generating Orthogonal Curvilinear Meshes", J. of Comp. Phys., Vol. 5, pp. 23-33, 1970
2) Joe F. Thompson, "General Curvilinear Coordinate Systems", in Numerical grid generation, pp. 1-30, Elsevier Sci. Pub. Co., Inc., (Ed) Joe F. Thompson, 1982
3) 권오준, "비정렬 격자를 이용한 3차원 천음속 팬 점성유동 해석", 한국항공우주학회지, 제26권, 제4호, 1998
4) 박영민, 권오준, "2차원과 3차원 비정렬 점성 격자 생성 기법 개발", 한국항공우주학회지, 제26권, 제4호, 1998
5) 이민철, 전만수, "3차원 곡면상의 사각형 요소망 자동생성을 위한 일반적 접근방법", 대한기계학회 추계학술 논문집 A, 1996, pp. 1160-1165
6) P. Devloo, J.T. Oden and P. Pattani, "An h-p Adaptive Finite Element Method for the Numerical Simulation of Compressible Flow", Computer methods in Applied Mechanics and Engineering, Vol. 70, No. 2, 1988, pp. 203-235
7) 김진환, "원심펌프 회전차 형상 설계에 대한 연구", 한국해양공학회지, 제 11권, 제 4호, 1997
8) G.F. Carey and J.T. Oden, "Finite Elements, A Second Course, Volume 2", Prentice-Hall, Inc., 1983, pp. 89-95.
9) Leon Lapidus and George F. Pinder, "Numerical Solution of Partial Differential Equations in Engineering", John Wiley & Sons, Inc., 1982, pp. 141-144
10) Rod Stephens, "Inside Secrets Programming VB Graphics", SamGakHyung Press, 1998
11) 강동진, 손정락, "병렬 컴퓨터에서 다중블록 유한체적법을 이용한 비압축성 유동해석", 유체기계저널, 제1권, 제1호, 1998