900 대한기계학회논문집(B) 제21권 제7호, pp. 900~910, 1997

〈논 문〉

하이퍼큐브++를 이용한 다중블록 격자생성

박 상 근* · 이 건 우**

(1996년 11월 13일 접수)

A Hypercube++ Approach for Multiblock Structured Grids

Sangkun Park and Kunwoo Lee

Key Words: Multiblock (다중블록), Structured Grids (구조화 격자), Hypercube++ (하이퍼큐브++), B-Spline Volume(비스플라인 부피)

Abstract

Multiblock structured grids are, to a large extent, capable of filling up topologically complex flow domains in an efficient way. The proposed approach enables to use different flow models in each different block and the easy incorporation of different grid refinement strategies for different blocks. Furthermore, it may be expected that this multiblock structured approach will naturally lead to the parallel executions of calculations per block on different vector processors. In this paper, the hypercube++ structure is proposed for topological informations on multiblock grids and the B-spline volume for geometrical informations. Three samples of the three dimensional results are presented to demonstrate the capabilities of the present approach.

1. 연구배경

전산유체를 위한 해석코드는 비교적 복잡한 기하학적 구조를 가진 비행기 주위의 물리영역을 해석하는 정도에까지 이르렀고, 보다 안정되고 빠른 해석코드를 개발하기 위해 계속적으로 연구 중에 있다. 현재 이러한 해석코드의 개발을 위해 가장 중점을 두고 있는 것은 계산영역의 생성이다. 즉 수치해석을 위한 모든 과정중에 임의의 물리영역에서 계산영역을 생성하는데 소모되는 시간과 경비가 가장 크다. 특히 3차원 문제의 경우에는 더욱 그러하다. 지난 수십년 간 계산영역을 생성하는 격자생성 코드는 오로지 하나의 블록(block)을 입력으로 단블록 격자(single-block grids)만을 생성해왔다. 이러한 단블록 해석코드와 단블록 격자는 비교적 단순한 형상에만 적용되고 또한 컴퓨터 메모리의 크기에 의해 제한받게 된다. 이러한 한계점을 극복하기 위해 다중블록이 등장하게 되었고 이를위해 다중블록 해석코드와 다중블록 격자생성에 관한 연구가 활발하게 진행되고 있다. 특히 다중블록 격자에 관한 연구[1~7]는 더욱 중요하다. 다중블록 격자란 해석하고자 하는 물리영역 즉 해석공간을 여러개의 작은 부영역(sub-domain)으로 나누어 각각의 부영역인 격자블록에 대해, 이웃한 격자블록과의 완만한 연결을 고려하여 생성한, 격자들의 집합으로서 격자블록 간의 상대적 위치관계 및 블록의 특성이 포함되어 있어야 한다. 그리고 다중블록 해석코드란 다중블록 격자를 입력으로 각 격자블록에 대해 적당한 경계조건을 가지고 반복계산에 의해 해석을 하면서 동시에 이웃한 격자블록과의 긴밀한 정보교환을 통해 전체 영역을 해석해 나가는 코드로서 종전의 단블록 해석코드에 비해 상당히 복잡한 구조를 가지고 있다.

*삼성 SDS 정보기술연구소 S/W응용개발팀
**서울대학교 기계설계학과

2. 다중블록의 필요성

이러한 다중블록 개념에 의한 영역분할은 다음과 같은 커다란 장점을 가지게 된다. 가장 먼저 기하학적 복잡성의 극복이다. 비구조화 격자(unstructured grid)에 비해 구조화 격자(structured grid)의 약점으로 등장하는 기하학적 복잡성 문제를 크게 해결할 수가 있다. 구조화 격자는 다소 복잡한 영역을 표현하는데 한계가 있다고 알려져 있지만 이는 단블록 격자를 가정했을 때의 상황으로서 다중블록 격자의 경우에는 반드시 그러하지는 않다.

두번째로 얻게 되는 장점은 단블록 격자에 비해 격자질의 향상을 꾀할 수 있다. 즉 단블록 격자에서 흔히 발생하는 치우침(skewness) 등을 막을 수 있어 격자의 직교성 및 완만성을 향상시킬 수 있다. 특히 직교성의 향상은 두드러지게 나타난다. 이러한 격자질의 향상은 궁극적으로 해석 상의 수치적 에러유발을 막게 되고 수치해의 빠른 수렴을 가져오게 하여 해석시간의 단축과 함께 보다 정확한 수치해를 얻을 수 있게 해준다.

세번째로 다중블록 해석코드에 의한 병렬처리 계산[8]을 가능하게 해 준다. 모든 관련된 수치적 계산을 단하나의 CPU를 가지고 순차적으로 계산한다면, 복잡한 구조를 가진 문제의 경우에 그 복잡성에 준하여 엄청난 격자점이 필요하게 되어 컴퓨터 메모리상에 커다란 문제를 일으키게 된다. 또한 엄청난 계산시간의 소모도 예측할 수 있다. 이러한 문제점들을 극복하기 위해 등장한 것이 병렬처리 계산기법으로서 하나의 가상 컴퓨터가 여러개의 CPU를 가지고 동시적으로 여러 명령을 수행하여 계산시간의 단축을 가져오게 한다. 이상의 다중블록 격자의 장점들은 장점인 동시에 안정되고 정확한 그리고 빠른 수치해석 결과를 얻기 위해 필요한 필요성이기도 하다.

3. 다중블록의 격자구조

복잡한 기하학적 구조를 가진 물리영역과 복잡한 물리현상 구조를 가진 문제를 안정되고 정확하게 해석하기 위해 도입한 다중블록 격자 개념을 구현하기 위해서 필요한 다중블록 격자의 구조는 다음과 같다. 크게 두 가지로 나누어 설명된다.

- 다중블록의 위상학적 구조 (topology structure)
- 다중블록의 기하학적 구조 (geometry structure)

다중블록의 위상구조란 각각의 격자블록 간의 상대적 위치관계를 정의하는 데이터 구조로서 이웃한 격자블록을 쉽게 찾아내어 이웃한 격자블록의 격자로부터 해석정보를 받거나 또는 이웃한 격자블록의 격자에게 해석정보를 전달할 수 있는 구조이어야 한다. 이는 해석과정의 관점에서 정보흐름을 윤활하게 해주는 매우 중요한 요소이다. 그리고 특정 영역의 위상구조를 결정할 때, 격자블록 간의 상대적 위치관계도 중요하지만 각 격자블록의 구조도 매우 중요하다. 매핑관계로부터 유도되는 인위적 특이성이 존재하지 않게끔 각 격자블록의 구조를 결정해야 한다. 그리고 가능하면 복잡한 물리현상이 나타나는 영역을 다른 영역과 구별하여 격자블록을 생성하고 그 물리현상의 특성을 수치해석코드에 입력으로 전달할 수 있다면 보다 안정된 해석결과를 기대할 수 있다. 한편 다중블록의 기하학구조란 실제 물리영역(해석공간)과 계산영역(매개영역)사이의 매핑관계를 정의하는 수학적 모델로서 편미분 관계식, 대수적 관계식, 매개변수 관계식 등에 의해 표현된다. 여기서 매핑관계식은 반드시 특정문제에만 적용되는 관계식이면 안된다. 일반화된 관계식 혹은 표현식이어야 한다. 또한 가능하면 매핑관계식이 쉽고 빠르게 물리영역을 표현하여 격자생성에 소모되는 시간을 없애야 하며, 가장 중요한 경계조건의 부여가 용이하도록 해야 한다. 그리고 전체적으로 혹은 국부적으로 격자밀도 혹은 격자간격을 용이하게 조정할 수 있어야 한다. 이러한 격자조정은 매우 중요하며 적응격자를 구현하거나 다격자(multigrid)에 의한 해석과정을 실현시키기 위해서 매우 필요한 요소이다.

본 연구에서는 이러한 다중블록 격자를 구현하기 위해서 다음과 같은 다중블록 격자구조를 제시한다. 다중블록 격자의 위상학적 구조로서 하이퍼큐브++(hypercube++)를 제시하고, 다중블록 격자의 기하학적 구조로서 매개변수 매핑관계식인 비스플라인 부피(B-spline volume)를 제시한다.

4. 비스플라인 부피

본 연구에서 구현한 B-spline부피[9]의 여러 관련

기능들에 대해서 그 자세한 내용은 생략하고 여기서는 정의 및 응용분야만을 간략히 소개하겠다. B-spline부피는 매개변수 u, v, w에 대한 벡터함수이며, 3차원인 유클리드 공간 상에서 다음 식으로 정의된다. 이 표현식은 B-spline곡선의 텐서곱(tensor product) 형태이다.

$$\boldsymbol{B}(u, v, w) = \sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} \boldsymbol{B}_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w) \quad (1)$$

여기서 $nu+1, nv+1, nw+1$은 각각 u, v, w방향으로의 조정점의 갯수이고, ku, kv, kw는 B-spline함수 $N_i^{ku}(u)$, $N_j^{kv}(v)$, $N_k^{kw}(w)$의 차수이다. 그리고 $\boldsymbol{B}_{ijk}$는 (i, j, k)번째의 조정점이고, $N_i^{ku}(u)$, $N_j^{kv}(v)$, $N_k^{kw}(w)$는 각각 절점벡터

$$\begin{aligned}
\boldsymbol{U} &= \{u_i\}_{i=0}^{nu+ku+1} \\
&= \{\bar{u}_0, \cdots, \bar{u}_0, \bar{u}_1, \cdots, \bar{u}_1, \cdots, \bar{u}_e, \cdots, \bar{u}_e\}, \\
\boldsymbol{V} &= \{v_j\}_{j=0}^{nv+kv+1} \\
&= \{\bar{v}_0, \cdots, \bar{v}_0, \bar{v}_1, \cdots, \bar{v}_1, \cdots, \bar{v}_f, \cdots, \bar{v}_f\}, \\
\boldsymbol{W} &= \{w_k\}_{k=0}^{nw+kw+1} \\
&= \{\bar{w}_0, \cdots, \bar{w}_0, \bar{w}_1, \cdots, \bar{w}_1, \cdots, \bar{w}_g, \cdots, \bar{w}_g\},
\end{aligned}$$

로 정의되는 B-spline함수이다. 이때 e, f, g는 u, v, w방향으로의 세그먼트 개수로서 각각 $nu-ku+2$, $nv-kv+2$, $nw-kw+2$와 같다.

본 연구에서 제시하는 B-spline부피 표현식은 비단 격자 생성에 응용될 뿐만 아니라, 세변수에 의해 표현 가능한 여러 분야로도 확장되어 응용가능하다. 예를들어 유동장의 속도분포, 온도분포, 압력분포 등을 표현하는데 응용될 수 있다. 뿐만 아니라 시간의 흐름에 따른 곡면의 거동 등을 모델링하는데도 응용될 수 있다.

5. 하이퍼큐브++

5.1 하이퍼큐브 관련연구

하이퍼큐브(hypercube)는 Allwright[3]에 의해 수치적 격자생성 분야에 처음으로 소개되었다. 그는 대상모델의 위상학적 구조와 그 모델 주위의 물리영역의 위상학적 구조를 시스템에서 제공하는 그래픽 도구를 사용하여 사용자가 직접 스케칭하는 과정을 소개하였고, 이와같은 그래픽 방식에 의한 위상구조 스케칭 방식의 경험을 토대로 여러가지 원칙과 전략을 유도해 낼 수 있다고 말했다. 여기서 그는 하이퍼큐브에 대해 간략히 소개하였고 대상모델의 어느 한 요소가 국부적으로 복잡한 구조를 가진 경우에 하이퍼큐브의 적용을 언급하였다. 그 후 Dannenhoffer[10]는 2차원에 대해서 실제형상과 같은 위상구조를 가지는 축약형상의 생성 방식에 관해 소개하면서 3차원으로의 확장성에 관해 언급하였다. 여기서 그는 바로 3차원으로의 확장은 매우 힘들며 Allwright의 하이퍼큐브를 도입하여 축약형상을 생성한다면 3차원으로의 확장은 가능하다고 언급하였다.

5.2 하이퍼큐브 구조

하이퍼큐브는 전체적으로 볼록한 형상을 한 임의의 모델을 감싸기에 매우 유용한 구조를 가지고 있다. 즉 비교적 단순한 형상을 가진 모델주위의 물리영역을 다중블록화 시키기에 매우 적당한 구조를 가지고 있다. 하이퍼큐브의 구조는 Fig. 1과 같다. 하나의 6면체(내부)가 존재하고 이 6면체를 둘러싸는 또다른 6면체(외부)가 존재하여, 이들 내부 및 외부 6면체는 각각의 대응 꼭지점에 의해 연결되어 있다.

본 연구에서는 이러한 하이퍼큐브 구조의 위상구조를 다음과 같이 간략히 데이터구조화 한다. 하나의 하이퍼큐브를 동(East)-서(West)-남(South)-북(North)-전(Front)-후(Back)-중(Center)의 위치관계를 가지는 7개의 영역(블록)으로 나눈다. 이때 7개의 블록은 다음과 같다. 내부 6면체에 중블록이 존재하고, 내부 6면체와 외부 6면체 사이에 동블록, 서블록, 남블록, 북블록, 전블록, 후블록이 존재하여 모두 7개의 블록이 존재한다. 이상의 구조로부터 각 블록 간의 상대적 위치관계는 자연스럽게 정의되고, 만약 중블록 안에 모델이 위치한다면 모델 주위의 영역은 동-서-남-북-전-후의 위상학적 위치관계를 가지는 6개의 블록에 의해 표현된다. 이상의 위상정보는, 단순한 형상을 가진 모델의 경우, 다중블록화에 필요한 위상학적 정보를

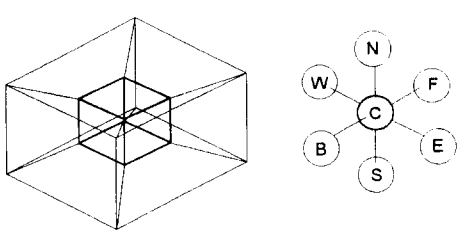


Fig. 1 Hypercube structure

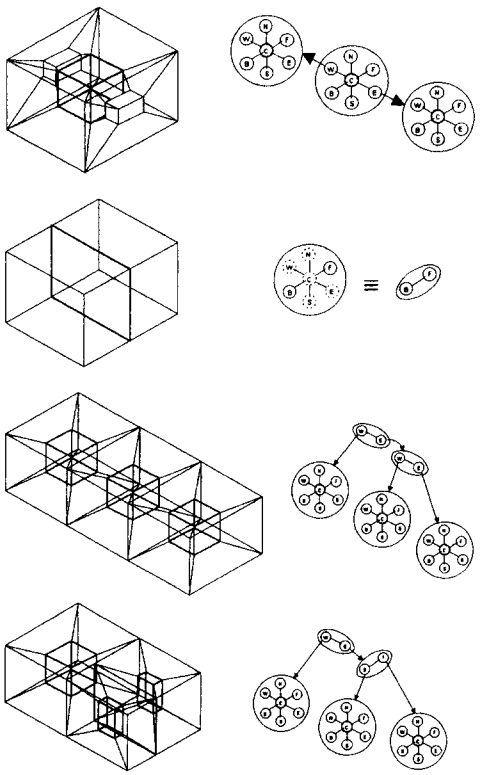


Fig. 2 Hypercube++ structure

충분히 제공하게 된다.

5.3 하이퍼큐브++ 구조

하이퍼큐브++의 기본틀은 기존의 하이퍼큐브를 바탕으로 하여 진일보한 확장된 형태의 구조를 가진다. 여기서 Fig. 2에서와 같이 확장된 형태의 구조란 하이퍼큐브를 구성하는 임의의 블록 내에 또 다른 하이퍼큐브를 위치시켜 부모-하이퍼큐브의 임의의 블록 내에 자식-하이퍼큐브가 존재하게 하는 계층구조를 말한다. 또한 중블록을 하나의 면으로 축약시켜 이 면을 기준으로 두개의 블록만이 하이퍼큐브 내에 존재하게 하는 변형된(degenerated) 하이퍼큐브의 구조를 말한다. Fig. 2의 오른쪽 그림은 왼쪽의 하이퍼큐브++의 데이터 구조를 알아보기 쉽게 표시한 것이다.

5.4 하이퍼큐브++ 다중블록화

5.4.1 다중블록화 전체과정

본 연구에서 제시하는 다중블록화 구현을 위한 기본 접근방식은 다음과 같다.

기본 접근방식

복잡한 공간구조를 한 실제 공간을 매우 단순한 구조를 가진 파라미터 공간(추상공간)으로 변환시키고, 생성된 파라미터 공간 상에서 필요한 작업을 수행한 후, 다시 원래의 실제 공간으로 재변환시켜 원하는 결과를 얻는다.

여기서 필요한 작업이란 물리공간을 분할하고, 분할된 공간 내에서 격자를 생성하는 작업을 말한다. 즉 다중블록에 의한 격자생성을 말한다. 그리고 파라미터 공간이란 본 연구에서 제시하는 매핑관계에 의해 복잡한 물리공간이 정형화된 혹은 추상화된 사각공간(squared or abstract space)으로 변환했을 때의 공간구조를 말한다. 한편 재변환이란 파라미터 공간에서 실제 물리공간으로의 매핑을 의미한다.

결국 본 연구에서 제시하는 매핑관계에 의해 격자생성의 대상인 복잡한 물리영역은 파라미터 공간 내에서 단순화되고, 단순화된 공간 내에서의 영역분할 및 격자생성 작업은 매우 빠르고 쉽게 처리되어진다. 현재까지의 다중블록과 관련된 연구 혹은 개발된 시스템이 쉽고 빠르게 그리고 자동적으로 다중블록화를 구현할 수 없었던 이유가 바로 복잡한 물리영역 내에서 물리공간을 설명하고 필요한 작업을 수행해 왔기 때문이다.

본 연구에서는 위에서 설명한 기본 접근방식을 구현하기 위해 다음의 데이터 구조 및 알고리즘을 제시한다.

Data Structure	Topology : hypercube++ Geometry : B-spline volume
Algorithms	Local Mapping : hypercube++ generation Domain Decomposition : hypercube++ merging

복잡한 물리공간의 다중블록화을 구현하기 위해, 데이터 구조로서 위상정보를 책임지는 하이퍼큐브++와 기하학 정보를 책임지는 비스플라인 부피를 제시하고, 구현 알고리즘으로서 매핑관계를 정의하는 하이퍼큐브++ 생성 알고리즘을 제시하고, 정의된 매핑관계에 의해 단순화된 파라미터 공간 내에서 영역분할 작업을 수행하는 하이퍼큐브++ 병

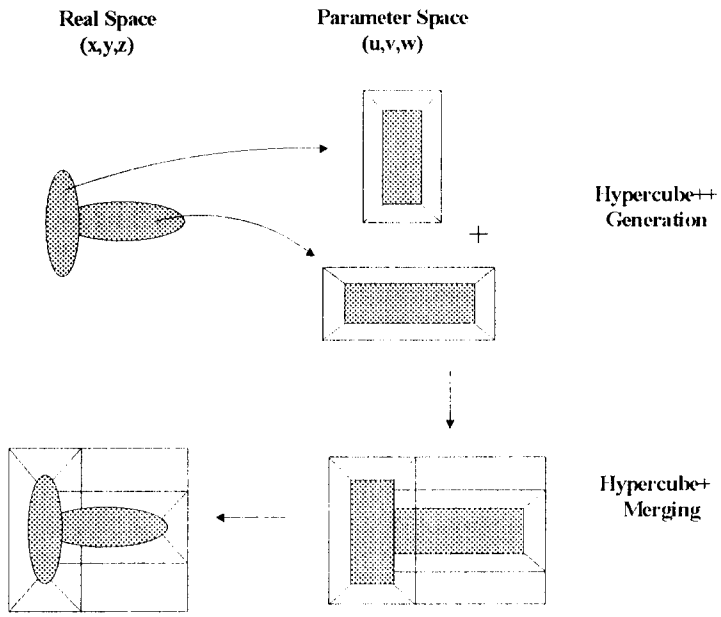


Fig. 3 Global steps of the suggested multiblock generation

합 알고리즘을 제시한다.

이상의 데이터 구조 및 구현 알고리즘을 가지고 다중블록화 과정을 순차적으로 나타내면 Fig. 3과 같다.

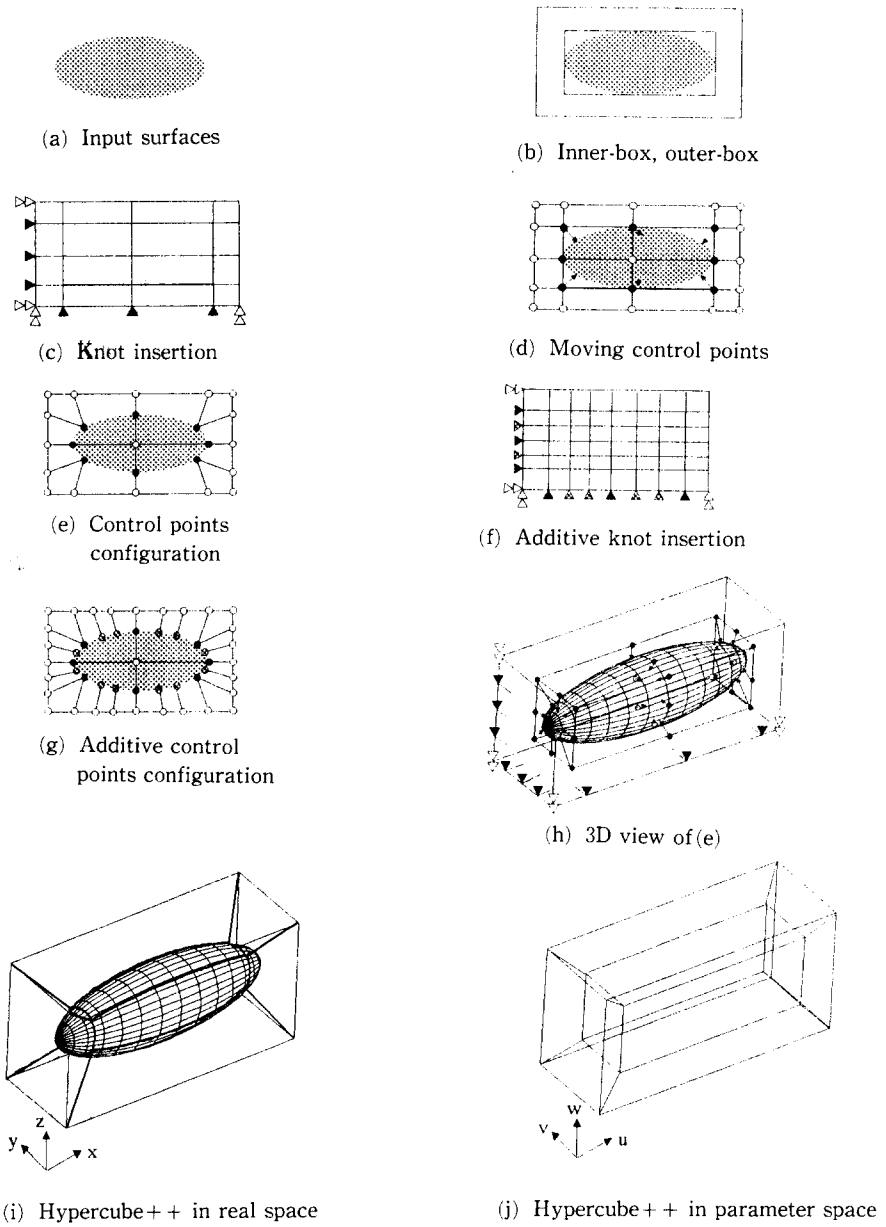
5.4.2 하이퍼큐브++ 생성

입력된 곡면집합 주위의 물리영역을 기본 하이퍼큐브에 의해 표현한다. 즉 기본 하이퍼큐브의 구성요소인 블록들에 의해 입력된 곡면집합 주위의 물리영역을 분할한다. 구체적인 생성과정을 살펴보면 Fig. 4와 같다.

먼저 입력된 곡면집합(Fig. 4(a))을 최소로 감싸는 내부상자(inner-box)를 생성하고, 이 내부상자를 중심으로 내부상자를 포함하는 적당한 크기의 외부상자(outer-box)를 생성한다. (Fig. 4(b)) 여기서 적당한 크기란 경계층의 크기 혹은 사용자 요구 등을 반영한 크기를 말한다. 이렇게 생성된 외부상자를 가지고 같은 크기의 비스플라인 부피를 생성한다. 그리고 내부상자의 최대점, 최소점, 중간점의 좌표값을 가지고 생성된 비스플라인 부피에 절점삽입(knot insertion)을 수행한다. (Fig. 4(c)) 그 다음 절점삽입에 의해 생성된 조정점들 중에서 입력된 곡면집합의 외부에 위치한 조정점들을 곡면집합과 가장 가까운 거리에 있는 곡면 상의 위치로 이동시킨다. (Fig. 4(d)) 이렇게 생성된 비스플라인 부피(Fig. 4(e), (h))는 부피내부에 입력된 곡면집합의 형상을 근사적으로 가지게 된다. 즉 생성된 비스플라인 부피를 내부상자의 최대점, 최소점의 좌표값을 가지고 절단해보면, 이 비스플라인 부피가 부피내부에 근사적으로 곡면집합의 형상을 가지고 있음을 알 수 있다.

그러나 이 부피는 곡면집합의 형상에 따라 정확도 면에서 떨어질 수 있다. 결국 좀더 정확성을 기하기 위해서 적당한 위치에 많은 수의 절점삽입(Fig. 4(f))을 통하여 많은 수의 조정점을 생성하고, 이 조정점들을 위에서 설명한 것과 같이 입력된 곡면집합과 가장 가까운 거리에 있는 곡면 상의 위치로 각각 이동시킨다. 여기서 적당한 위치란 내부상자 내부를 말하는데 본 연구에서는 내부상자를 균등하게 분할하는 위치로 정의했다. 이렇게 생성된 비스플라인 부피(Fig. 4(g))는 이전의 부피에 비해 많은 수의 조정점을 가지고 좀더 정확하게 부피 내부에서 곡면집합의 형상을 근사하게 된다.

이상의 과정을 실제 물리공간이 아닌 비스플라인 부피의 파라미터 공간에서 보면, 입력된 곡면집합은 내부상자의 형상(Fig. 4(j))으로 나타난다. 즉 복잡한 형상이 파라미터 공간에서 직육면체 모양을 한 상자모양으로 축약된다. 그리고 곡면집합 주위의 물리영역은 파라미터 공간에서 외부상자에서 내부상자를 뺀 영역으로 단순화된다. 이와같이 생성



(a) Input surfaces

(b) Inner-box, outer-box

(c) Knot insertion

(d) Moving control points

(e) Control points configuration

(f) Additive knot insertion

(g) Additive control points configuration

(h) 3D view of (e)

(i) Hypercube++ in real space

(j) Hypercube++ in parameter space

Fig. 4 Hypercube++ generation

된 비스플라인 부피는 복잡한 공간구조를 한 물리영역과 단순한 직육면체 모양을 한 파라미터 공간 사이에서 일대일 대응관계를 맺어주는 매핑함수로서 그 역할을 수행한다.

그 다음 과정은 파라미터 공간 상에서 기본 하이퍼큐브 생성을 통하여 영역분할을 수행하는 것이다. (Fig. 4(i), (j)) 내부상자를 생성할 하이퍼큐브의 중블록에 위치시키고, 내부상자의 각 꼭지점을 외부상자의 대응하는 꼭지점에 연결시켜 나머지 블록들을 생성한다. 이 과정은 매우 간결해서 파라미터 공간 상에서 쉽게 영역분할을 수행하게 된다.

이상의 하이퍼큐브++ 생성과정을 통하여 작업공간이 복잡한 물리공간에서 단순한 파라미터 공간으로 이동하며, 이 파라미터 공간 상에서 영역분할

을 수행하고, 필요에 따라 분할된 각 영역을 다시 물리공간으로 이동시켜 최종의 원하는 블록들을 생성한다. 이러한 하이퍼큐브++ 생성과정은 본 연구에서 자동으로 수행된다.

5.4.3 하이퍼큐브++ 병합

파라미터 공간 상에서, 입력된 두 하이퍼큐브++는 하이퍼큐브++ 병합작업에 의해 하나의 하이퍼큐브++로 합쳐진다. 또한 합쳐진 하이퍼큐브++ 간에도 다시 하이퍼큐브++ 병합작업에 의해 하나의 하이퍼큐브++로 합쳐진다. 이러한 하이퍼큐브++ 병합작업을 통하여 임의의 개수의 하이퍼큐브++는 좀더 복잡한 데이터 구조를 가지는 하나의 하이퍼큐브++로 그들의 상대적 위치에 제한없이 합쳐진다. 여기서 상대적 위치에 제한없음이란 입력된 두 하이퍼큐브++가 겹치지 않고 떨어져 있든, 혹은 완전히 겹쳐 포함되어 있든, 또는 두 하이퍼큐브++가 부분적으로 겹쳐져 있든, 그들 간의 겹침관계에 아무런 제한을 받지 않음을 의미한다. 이와같은 하이퍼큐브++ 병합작업을 통하여 주어진 곡면모델 주위의 물리영역은 하나의 하이퍼큐브++에 의해 표현된다. 즉 생성된 하이퍼큐브++로부터 최종적으로 원하는 영역분할을 실현하게 된다.

한편, 하이퍼큐브++ 병합작업의 전체구조를 살펴보면 Fig. 5와 같다. 입력된 두 하이퍼큐브++ 간의 상대적 위치관계에 따라 크게 3가로 나누어 설명된다. Fig. 5의 (a)는 두 하이퍼큐브++가 겹쳐있지 않고 떨어져 있는 경우이며, (b)는 완전히 겹쳐 포함되는 경우이며, (c)는 서로 부분적으로 겹쳐져 있는 경우이다. 각 경우에 대해 적용되는 알고리즘 및 적용결과는 그림과 같다. 먼저 (a)의 경우, outer-merge알고리즘이 적용되어 두 하이퍼큐브++가 차지하는 영역을 하나의 하이퍼큐브++로 표현한다. (b)의 경우는 inner-merge알고리즘에 의해 포함되는 하이퍼큐브++가 포함하는 하이퍼큐브++ 안에 흡수되어 합쳐지게 된다. 물론 합쳐진 결과는 하나의 하이퍼큐브++가 되어, 두 하이퍼큐브++가 차지하는 영역을 표현하게 된다. (c)의 경우, 먼저 hypercut 오퍼레이터가 적용되어 하나의 하이퍼큐브++를 상대 하이퍼큐브++의 외곽경계면을 절단면으로 해서 여러 개의 조각으로 자른다. 잘려진 조각들도 물론 하이퍼큐브++의 구조를 가지게 된다. 이렇게 잘려진 조각들 중에, 먼저 상대 하이퍼큐브++의 내부공간에 위치하는 하이퍼큐브++는 상대 하이퍼큐브++와 함께 inner-merge알고리즘에 의해 합쳐지게 되고, 그 다음 합쳐진 하이퍼큐브++는 나머지 잘려진 하이퍼큐브++ 조각들과 함께 outer-merge알고리즘에 의해 다시 합쳐지게 되어, 결국 입력된 두 하이퍼큐브++가 차지하는 영역을 하나의 하이퍼큐브++로 표현한다. 이상의 과정은 overlap-merge 알고리즘에 의해 구현된다. 여기서 알고리즘 outer-merge, inner-merge, overlap-merge 및 오퍼레이터 hypercut에 관한 자세한 내용[11]은 생략한다.

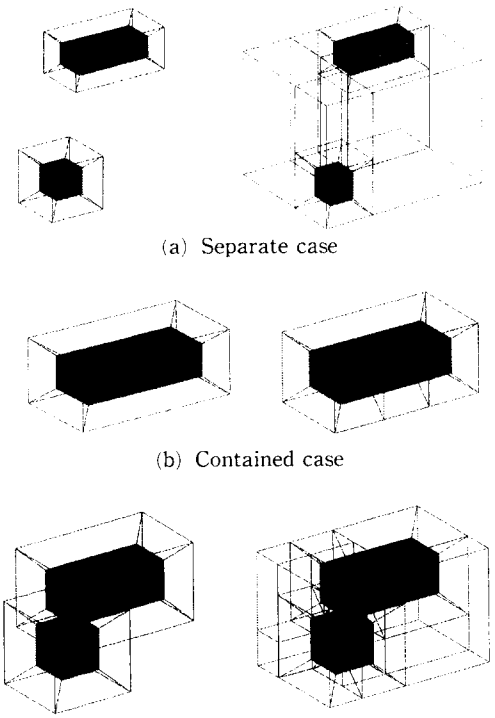


Fig. 5 Hypercube++ merging

5.4.4 하이퍼큐브++ 다중블록의 특징

본 연구에서 제시한 하이퍼큐브++ 데이터 구조와 기본 접근방식에 기초하여 구현된 다중블록화 생성방식의 특징은 다음과 같이 정리된다.

● 기존의 사용자 경험에 의존한 그래픽 접근방식에 비해 영역분할 과정에 소요되는 시간과 노력이 상당히 저렴하며, 다중블록의 위상구조가 자동적으로 생성되어 사용자가 그래픽 도구를 사용하여 직접 위상정보를 생성하거나 혹은 수정할 필요가

없다.

● 자주 사용되는 특징형상에 관하여 해당하는 하이퍼큐브++를 저장해 놓았다가 필요시 재사용 가능하다. 이러한 재사용을 통하여 특징형상에 기반을 두고 모델링 작업을 수행하는 솔리드 모델링 시스템의 작업 구조를 구현할 수 있다.

● 하이퍼큐브의 확장개념인 하이퍼큐브++ 계층구조를 기반으로 구현한 하이퍼큐브++ 병합 알고리즘은 시스템 내부에서 비교적 쉽게 블록들 간의 위상학적 위치관계를 정의할 수 있고, 또한 생성된 하이퍼큐브++ 구조로부터 또한 모델에 인접한 블록들을 생성된 하이퍼큐브++ 구조로부터 추가적인 계산없이 자동적으로 찾아낼 수 있다.

● 생성된 하이퍼큐브++ 계층구조로부터 비교적 쉽게 이웃블록에 관한 정보를 추출해 낼 수 있다. 이웃블록에 관한 정보는 유동해석 시에 이웃한 블록들 간의 유동해석 데이타의 교환 혹은 격자생성 시에 블록들 간의 경계면에서의 격자점 일치 등을 위하여 매우 필요하다.

● 물체 표면과 접촉하고 있는 블록들을 생성된 하이퍼큐브++ 구조로부터 비교적 쉽게 찾아낼 수 있다. 일반적으로 물체 표면과 가까운 거리에 있는 영역들에 대한 유동해석은 매우 중요하며, 이를 위하여 흔히 물체 표면 근처에서 격자의 직교성 혹은 격자의 해상도를 증가시킨다.

● 주어진 위상구조에 새로운 특징형상의 요소를 첨가함으로써 변화하는 위상구조에 대하여 이미 생성된 다중블록을 없애고 처음부터 다시 재생성할 필요가 없다. 기존의 그래픽 중심의 방식은 이러한 첨가된 위상구조에 국부적으로 대응하지 못한다. 일반적으로 고수준의 성능향상을 위하여 새로운 특징형상 요소를 흔히 첨가하며 이에 따라 매번 재생성을 통하여 대응한다면 엄청난 시간과 노력의 낭비를 불러 일으키게 된다. 본 연구의 하이퍼큐브++ 병합과정은 이러한 재생성의 문제를 해결할 수 있다. 즉 국부적으로 추가된 위상구조에 대하여, 마치 솔리드 모델러에서의 블리안 작업과 같이, 추가된 위상구조에 해당하는 하이퍼큐브++를 생성한 후, 병합 알고리즘에 의하여 기존의 하이퍼큐브++에 병합시킴으로써 추가된 위상구조에 대한 문제를 쉽게 해결한다.

● 실제모델의 국부적 형상변화에 국부적으로 대응한다. 즉 기하학적 형상변화가 있는 영역의 블록들을 자동으로 찾아내어 국부적으로 형상변화에 적응한다.

● 본 연구에서 제시한 매핑함수에 의한 작업공간의 이동을 통하여 작업공간이 복잡한 공간구조한 물리공간에서 육면체 모양을 한 단순한 구조의 파라미터 공간으로 이동함으로써 보다 쉽고 빠르게 특정 알고리즘을 구현할 수 있다. 여기서 특정 알고리즘이란 영역분할에 필요한 작업을 말한다. 뿐만 아니라 격자생성에 필요한 기타 작업도 여기에 속할 수 있다. 이러한 작업공간의 이동은 비단 다중블록화의 구현뿐만 아니라 복잡한 물리공간에서 특정한 기하학적 문제를 해결해야 하는 분야 등에 응용될 수 있다.

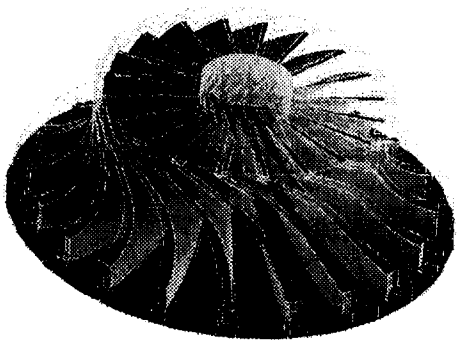
● 하이퍼큐브++ 병합작업의 관점에서, 실제모델의 갯수 및 상대적 위치는 의미가 없다. 즉 복잡한 구조를 분할없이 전체적으로 해결하려는 기존의 접근방식과는 다르게, 주어진 위상구조에 대해 모델의 특징형상을 단위로 각각의 하이퍼큐브++를 생성하고, 그들을 3차원 공간 상에서 그들의 위치관계에 무관하게 합쳐나가는 본 연구방식은 주어진 모델의 갯수가 하나이든 여러개이든 그리고 모델들이 겹쳐있든 멀리 떨어져 있든 제한없다.

● 모델을 구성하는 곡면들 간의 인접여부에 비교적 관대하다. 즉 곡면들간의 연속성 조건을 특별히 요구하지 않는다. 이러한 특징은 비전문가도 쉽게 모델 생성에 참여할 수 있게 해준다.

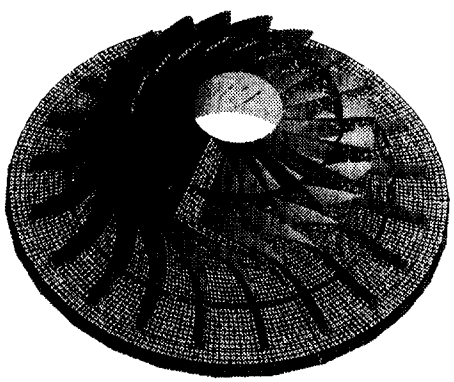
● 본 연구방식은 사용할 격자 생성 방식에 무관하다. 즉 생성하고자 하는 격자의 형태가 블록 구조화 격자, 비구조화 격자, 혹은 하이브리드 격자이든 무관하게 현존하는 상용 시스템에 적용될 수 있다. 일반적으로 복잡한 구조에 대한 격자 생성을 위하여 생성할 격자의 형태가 무엇이든지 가장 먼저 해결해야 할 문제는 바로 영역분할 문제이다.

6. 적용예 및 결론

본 연구에서 제시한 하이퍼큐브++ 구조에 기초한 다중블록 생성방식의 적용 예제들을 살펴보면 다음과 같다. 이 예제들을 통하여 다중블록 영역분할 방식으로서의 하이퍼큐브++ 계층구조의 접근타당성을 확인할 수 있고, 또한 복잡한 3차원 물리영역에서의 가능성을 확인할 수 있다. 본 연구에서 적용한 예제는 굴곡이 심한 임펠러 구조, 복잡한 위상구조를 가진 항공기, 그리고 43개의 구조물로 이루어진 서울대학교 공대건물 단지이다.

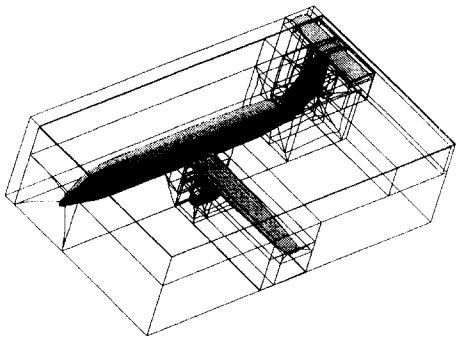


(a) Multiblock decomposition

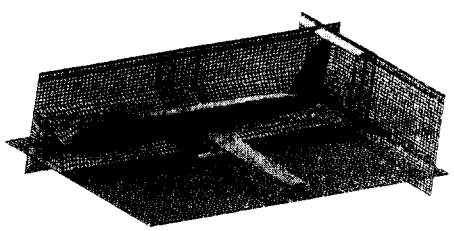


(b) Block-structured grids

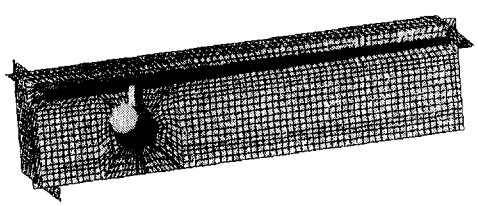
Fig. 6 Application of the hypercube++ approach to an impeller configuration



(a) Multiblock decomposition



(b) Block-structured grids



(c) Block-structured grids over wing-nacelle geometry

Fig. 7 Application of hypercube++ approach to an airplane configuration

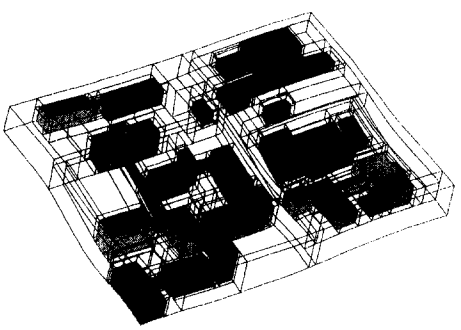
Fig. 6은 첫번째 예제로서 본 연구에서 제시한 하이퍼큐브++ 다중블록 생성방식을 비교적 굴곡이 심한 곡면들로 이루어진 임펠러 구조에 적용시켜, 생성된 영역분할의 결과와 이로부터 생성한 초기격자의 단면을 보여주고 있다. 그 과정을 간략히 살펴보면 다음과 같다. 먼저 12개의 날개로 이루어진 임펠러 구조에 대해 각 날개에 하이퍼큐브++ 생성 알고리즘을 적용시켜 12개의 기본 하이퍼큐브들을 생성하고, 하이퍼큐브++ 병합 알고리즘에 의해 하나의 하이퍼큐브++로 병합한다. 그다음 병합된 하이퍼큐브++가 제공하는 매핑함수인 비스플라인 부피의 파라미터 공간상에서 격자점을 생성한 후, 그 결과를 실제 물리공간으로 매핑함으로써 초기격자를 생성한다. Fig. 6(a)는 140개의 블록으로 이루어진 병합된 하이퍼큐브++를 보여주고 있다. 이를 통하여 본 연구방식에 의해 영역분할된 다중블록의 구조를 확인할 수 있다. 그리고 Fig. 6 (b)는 이를 바탕으로 생성시킨 블록 구조화 격자들의 특정 단면들을 보여주고 있다. 여기서 생성시킨 격자의 크기는 전체적으로 50×16×240이다.

두번째 예제로서 비행기 몸체(fuselage), 주날개(main wing), 나슬(nacelle), 꼬리날개(tail wing) 등으로 구성된 항공기에 대하여, 본 연구방식의 적용 결과가 Fig. 7에 나타나 있다. 6개의 특징형상 구조물로 이루어진 항공기의 각 구조물에 대해, 앞의 임펠러 예제와 같이, 하이퍼큐브++ 생성 알고리즘을 적용시켜 6개의 기본 하이퍼큐브들을 생성하고, 하이퍼큐브++ 병합 알고리즘에 의해 하나의 하이퍼큐브++로 병합한다. 이때 소요된 계산시간은 공학용 워크스테이션에서 대략 3분 정도이다. Fig. 7(a)는 157개의 블록으로 이루어진 병합된 하이퍼큐브++를 보여주고 있고, Fig. 7(b)는

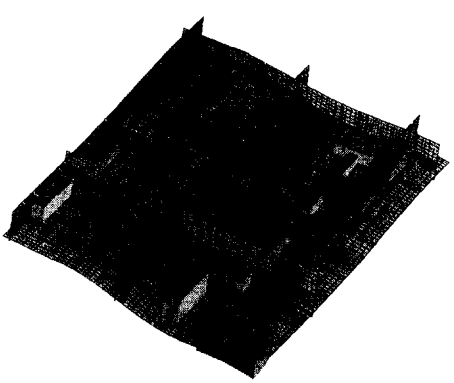
격자크기가 전체적으로 80×30×50인 블록 구조화 격자들의 특정 단면들을 보여주고 있으며, Fig. 7 (c)는 주날개와 나슬이 위치하고 있는 부위에서 블록 구조화 격자들의 특정 단면들을 자세히 보여주고 있다.

마지막으로 43개의 구조물로 이루어진 서울대학교 공대건물 단지에 대하여, 본 연구방식의 적용결과를 살펴보면 Fig. 8과 같다. 앞의 적용 예제들과 같이, 각 건물의 구조물에 대해 하이퍼큐브++ 생성 알고리즘을 적용시켜 43개의 기본 하이퍼큐브들을 생성하고, 하이퍼큐브++ 병합 알고리즘을 적용시켜 하나의 통합된 하이퍼큐브++로 병합한다. Fig. 8(a)는 304개의 블록으로 이루어진 병합된 하이퍼큐브++의 구조를 보여주고 있고, Fig. 8(b)는 격자크기가 전체적으로 70×50×10인 블록 구조화 격자들의 특정 단면들을 보여주고 있다.

전산유체 분야에서 가장 많은 시간을 요하는 부분은 격자를 생성하는 과정이고, 격자생성 과정에서 가장 어려운 부분은 바로 영역분할 과정이다.



(a) Multiblock decomposition



(b) Block-structured grids

Fig. 8 Application of hypercube++ approach to a building complex configuration

그리고 지금까지 블록 간의 위상정보를 자동적으로 생성해 내는 진정한 의미의 영역분할 방식은 없었다. 사용자가 시스템에서 제공하는 그래픽 도구 등을 사용하여 힘들게 영역분할을 수행해 왔다. 이는 영역분할 작업을 복잡한 실제 물리공간 상에서 구현하려는 점에 문제가 있기 때문이다. 또한 복잡한 형상의 모델을 복잡한 구조 그대로 다루려는 점에 문제가 있기 때문이다. 본 연구의 기본 접근방식은 이러한 기존방식의 복잡성을 해결할 수 있는 방향을 제시하며, 실제 하이퍼큐브++ 방식에 의해 작업공간의 복잡성과 모델의 복잡성을 해결해 준다.

후 기

본 연구는 1995년도 교육부 학술연구조성비(기계공학연구)에 의하여 연구되었음

참고문헌

(1) Thompson, J. F., 1996, A Reflection on Grid Generation in the 90s : Trends, Needs, and Influences, *Proceedings of the 5th International Grid Conference*, Mississippi, USA, pp. 1029~1110.

(2) Weatherill, N. P. and Forsey, C. R., 1984, Grid Generation and Flow Calculations for Complex Aircraft Geometries Using A Multi-Block Scheme, *AIAA Paper* 84~1665.

(3) Allwright, S. E., 1988, Techniques in Multiblock Domain Decomposition and Surface Grid Generation, *Proceedings of the 2nd International Grid Conference*, Miami, USA, Pineridge Press, pp. 559~568.

(4) Steinbrenner, J. P., Chawner, J. R. and Fouts C. L., 1989, A Structured Approach to Interactive Multiple Block Grid Generation, *AGARD FDP Specialist Meeting* in Loen, Norway, May.

(5) Arabshahi, A. and Whitfield, D. L., 1989, A Multi-Block Approach to Solving the Three-Dimensional Unsteady Euler Equations About a Wing-Pylon-Store Configuration, *AIAA Paper* 89~3401.

(6) Sorenson, R. L. and McCann, K. M., 1991, A Method for Interactive Specification of Multiple-Block Topologies, *AIAA*-91-0147.

(7) Shevare, G. R., et. al., 1996, Multi-Block Generation for Structured Grids in Volumes, *Proceedings of the 5th International Grid Conference*, Mississippi, USA, pp. 469~478.

(8) Geist, G. A. and Sunderam, V. S., 1992, Network Based Concurrent Computing on the PVM System, *Journal of Concurrency : Practice and Experience*, Vol. 4, No. 4, pp. 293~311.

(9) Casale, M. S. and Stanton, E. L., 1985, An Overview of Analytic Solid Modeling, *IEEE Comp. Graph. & Appl.*, Vol. 5, No. 2, pp. 45~56.

(10) Dannenhoffer, J. F., 1993, A New Method for Creating Grid Abstractions for Complex Configurations, *AIAA*-93-0428, AIAA 31th Aerospace Science Meeting, Reno, NV.

(11) 박상근, 1997, 다중블록 영역분할의 자동생성 및 그 응용, 박사학위 논문, 서울대학교.