효율적인 옷감 애니메이션 및 충돌 처리 기법

강영민
한국전자통신연구원 가상현실연구부
e-mail : ymkang@etri.re.kr

Efficient Methods for Cloth Animation and Collision Handling

*Young-Min Kang
Virtual Reality Research Department, ETRI

Abstract

This paper proposes efficient cloth animation and collision handling methods. There have been various techniques for the generation of cloth behavior. However, the cloth animation is still a challenging subject in real-time environments. This paper presents an efficient animation method based on implicit integration. The proposed method can efficiently animate virtual cloth object with complex geometry. In addition, this paper also introduces an efficient collision handling method. The collision resolution is another important issue in cloth animation since deformable objects has special collision problem called self-collision. In this paper, the self-collision was successfully avoided in real-time environments

I. 서론

많은 연구자들이 옷감의 정확한 움직임을 재현하려는 다양한 기법들을 제안하였지만 실시간 옷감 애니메이션은 여전히 어려운 문제이다 [3,6,11,12]. 효율적인 옷감 애니메이션을 위한 다양한 연구들 가운데 가장 중요한 최근의 성과는 암시적 적분법(implicit integration)을 이용하는 것이다. 그러나 암시적 적분법은 선형시스템의 풀이를 요구한다[1]. 이 선형 시스템은 다행히 희소행렬로 표현되지만, 복잡한 모델은 실시간 처리가 불가능하다. 몇몇 연구자들이 계산 부담을 줄일 수 있는 효율적인 기법들을 제안하였지만 [5,7], 이러한 기법들은 실시간 성능을 얻기 위해 정확성을 희생하였다. 몇 가지 기하적 기법을 이용한 방법이 제안되었지만 [4,9], 이는 물리적 모델에 기반을 두지 않은 모델이기 때문에 사실적인 동작을 생성하지 못하였다.

II. 애니메이션 모델

실시간 옷감 애니메이션의 가장 중요한 문제는 시스템의 안정성인데, 이는 암시적 적분법에 의해 보장될 수 있다. 따라서 질량 스프링 모델에 기반을 둔 옷감 모델의 동작을 다음과 같은 역 오일러(backward Euler) 기법을 통해 안정적으로 생성할 수 있다 [8]:

$$\begin{pmatrix} \mathrm{v}^{t+h} \\ \mathrm{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathrm{v}^{t} + h\mathrm{M}^{-1}\mathrm{f}^{t+h} \\ \mathrm{x}^{t} + h\mathrm{v}^{t+h} \end{pmatrix} \quad (1)$$

식 1에서 h는 시간 간격을 의미하며, v는 질점들의 속도를 원소로 하는 벡터이며, f와 x는 각각 질점들의 힘과 위치를 원소로 하는 벡터이다. 그리고 행렬 M은 질량 행렬이다. 위첨자 t와 $t+h$는 시간을 나타내는데, 의미적으로 현재 시간 상태와 다음 상태를 의미한다. 이렇게 표현된 수치적분의 계산 목표는 다음 상태의 위치, x^{t+h}를 찾는 것으로, 이는 속도의 변화를 계산하면 된다. 따라서 옷감 애니메이션은 결국 다음과 같은 속도 변화를 얻는 문제이다:

$$\Delta\mathrm{v}^{t+h} = h\mathrm{M}^{-1}\mathrm{f}^{t+h} \quad (2)$$

질량 스프링 기반의 애니메이션이 식 2와 같이 간단한 식으로 표현될 수 있지만, 다음 상태의 힘인 f^{t+h}를 정확히 알 수 있는 방법이 없어 다음과 같이 근사를 사용해야 한다.

$$\mathrm{f}^{t+h} = \mathrm{f}^{t} + \frac{\partial \mathrm{f}}{\partial \mathrm{x}}\Delta\mathrm{x}^{t+h} = \mathrm{f}^{t} + \mathrm{J}\,\Delta\mathrm{x}^{t+h} \quad (3)$$

이때 J는 힘 벡터를 각 정점의 위치로 이루어진 벡터 x로 미분한 자코비안(Jacobian) 행렬이며, 위치 변화 $\Delta\mathrm{x}^{t+h}$는 $\mathrm{v}^{t} + h\Delta\mathrm{v}^{t+h}$로 나타낼 수 있기 때문에, 질량 스프링 모델의 애니메이션은 결국 다음과 같은 선형 시스템을 푸는 문제가 된다.

$$(\mathrm{M} - h^2\mathrm{J})\Delta\mathrm{v}^{t+h} = h\mathrm{f}^{t} + h^2\mathrm{J}\,\mathrm{v}^{t} \quad (4)$$

일반적으로 식 4에 포함된 행렬 $\mathrm{M}-h^2\mathrm{J}$는 매우 큰 행렬이다. 다행히 이 행렬이 희소 행렬이기는 하지만, 복잡한 모델이라면 실시간 애니메이션이 어렵다.

III. 실시간 애니메이션

이 장에서는 식 (4)의 근사해를 효율적으로 구해 실시간 애니메이션을 얻는 기법을 설명한다. 설명을 단순히 하기 위해 $\mathrm{M}-h^2\mathrm{J}$를 간단히 W로 표현하자. 행렬 W는 매우 중요한 여러 특성을 가지며, 이러한 특성들을 적절히 이용함으로써 실시간 애니메이션이 가능하다.

식 4의 우변은 $h(\mathrm{f}^t+h\mathrm{J}\mathrm{v}^t)$로 표현될 수 있으며, 추가적인 힘 $h\mathrm{J}\mathrm{v}^t$는 점성력이다[5]. 자코비안 행렬이 희소행렬이라는 특성 때문에, 이 점성력을 간단히 계산할 수 있으며, 원래의 스프링 힘과 이 점성력의 합을 전체 내부힘 $\tilde{\mathrm{f}}$로 표현하면 식 4의 선형 시스템은 다음과 같이 간단히 표현될 수 있다:

$$\mathrm{W}\Delta\mathrm{v}^{t+h}=h\tilde{\mathrm{f}}^{t} \tag{5}$$

식 5의 선형 시스템은 n 개의 방정식으로 표현 가능한데, 이 가운데 질점 i의 속도변화가 좌변에 놓이는 방정식을 보면 다음과 같이 표현될 수 있다:

$$\Delta\mathrm{v}_i^{t+h}=\mathrm{W}_{ii}^{-1}h\tilde{\mathrm{f}}_i^{t}+h^2\sum_{(i,j)\in E}\mathrm{J}_{ij}\Delta\mathrm{v}_j^{t+h} \tag{6}$$

본 논문에서 제안하는 방법은 식 6에 나타난 속도변화 갱신식에 따라 반복적으로 갱신함으로써 선형 시스템의 해를 근사하는 것이다. 사실 새로운 첨자 k를 도입하여 반복 회수를 나타낸다면, 식 6은 자코비(Jacobi) 반복법과 동일하다. 즉 $\Delta\mathrm{v}_i^{t+h^{(k)}}$가 k 번의 반복에 얻어진 근사해라고 하고, 초기값을 $\Delta\mathrm{v}_i^{t+h^{(0)}}=\mathrm{W}_{ii}^{-1}h\tilde{\mathrm{f}}_i^{t}$ 라고 하면, 반복적 갱신 방법은 다음과 같이 얻어진다.

$$\Delta\mathrm{v}_i^{t+h^{(0)}}=\mathrm{W}_{ii}^{-1}h\tilde{\mathrm{f}}_i^{t}$$

$$\Delta\mathrm{v}_i^{t+h^{(k+1)}}=\mathrm{W}_{ii}^{-1}\left(h\tilde{\mathrm{f}}_i^{t}+h^2\sum_{(i,j)\in E}\mathrm{J}_{ij}\Delta\mathrm{v}_j^{t+h^{(k)}}\right) \tag{7}$$

식 7을 이용하여 적은 수의 반복을 수행하면, 옷감 모델의 움직임을 얻기 위한 선형 시스템의 근사해를 실시간에 얻을 수 있다. 본 논문의 반복 기법이 암시적 적분법에 기반을 두기 때문에, 애니메이션 결과는 실시간 애니메이션 시스템에 적용하기에 충분할 정도로 안정적이다. 실험을 통해 단 한 번의 반복을 통해 얻은 결과도 안정적이며 사실적인 동작을 보여주었다.

앞서 우리는 이 힘의 미분치를 사용했는데, 이 힘은 훅(Hooke)의 법칙으로 쉽게 계산할 수 있고, 그 미분치도 구할 수 있다. 그런데, 이 힘의 미분치가 안정성을 깨드릴 수 있다.

즉, 두 질점 i와 j가 서로 가까워지면, 이 미분치 행렬의 원소 값들은 무한히 커질 수 있다는 것이다. 이를 해결하기 위해 본 논문의 기법은 스프링 수축 시에 스프링의 휴지 길이가 언제나 현재 스프링 길이와 동일하다고 가정한 미분치를 사용하였다. 이를 이용함으로써 언제나 안전한 다음 상태 힘을 이용할 수 있었고, 실험 결과 이렇게 계산된 힘의 미분치는 옷감 애니메이션의 사실성을 크게 떨어뜨리지 않았다.

Ⅳ 효율적인 충돌 처리

애니메이션을 수행할 때, 물체의 외형이 변형 되는가에 관계없이 모든 형태의 객체에 대해 처리해야 하는 충돌은 애니메이션 객체와 외부 객체와의 충돌이다. 이러한 일반적 충돌은 다양한 계층적 방법을 통해 효과적으로 처리할 수 있는 방법이 많이 제시되어 있으며, 실시간 처리에 큰 문제가 발생하지 않는다. 따라서 본 논문에서는 객체가 지면이나 다른 객체와 충돌하는 경우에 대한 설명은 생략하도록 한다.

변형 가능한 객체의 동작을 생성할 때, 가장 큰 문제가 되는 충돌은 객체의 일부가 자기 자신의 다른 부분에 부딪히는 자체충돌(self-collision)이다. 이 자체 충돌 문제는 오랫동안 옷감과 같이 변형이 되는 물체의 동작을 생성하는 분야에서 동작 생성 시뮬레이션과 함께 중요한 계산상의 부담을 가져오는 문제로 알려져 있다. 자체 충돌에 관한 대표적인 논문은 볼리노(Volino)와 프로보트(Provot)에 의해 각각 발표되었다[10,13]. 볼리노는 옷감과 같이 변형 가능한 객체를 이루는 면들이 가진 모든 법선 벡터 N_i에 대해 $\mathrm{N}_i\cdot\mathrm{u}>0$인 벡터 u가 존재하면 자체 충돌이 없다는 것을 증명하였고, 이를 이용하여 자체 충돌 검사 횟수를 크게 줄인 기법을 제안하였다. 이 기법에 영향을 받아 프로보트는 각 면들이 가진 법선 벡터들의 합집합을 원뿔 형태로 저장하여 계층 구조를 구성하는 방법을 제안하였다. 이러한 방법은 어떤 노드의 하부 노드에 있는 법선 벡터들이 이루는 각도가 정해진 임계치 이하인 경우에는 검사를 하지 않아도 된다는 장점을 가지고 있지만, 긴 리본이 나선형으로 감겨 있는 경우 등에서는 불필요한 검사를 수행해야 한다.

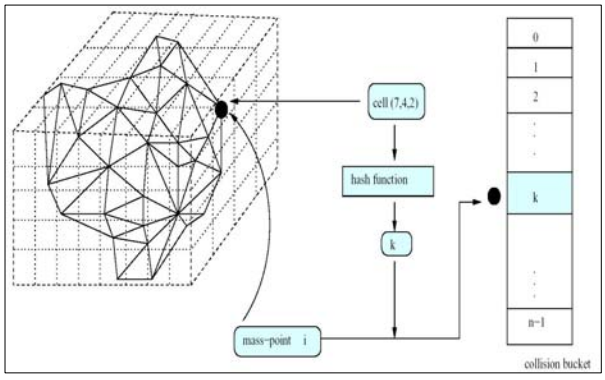

그림 1. 특정 정점을 버켓에 할당하는 방법

본 논문에서 제안하는 기법은 빠른 충돌 처리를 위하여 자체 충돌이 실제로 일어나는 것을 감지하여 반

응하는 것이 아니라 실시간에 회피하는 기법을 사용한다. 이 방법은 우선 그림 1과 같이 객체 전체의 바운딩 박스를 구하는 것으로 시작한다. 바운딩 박스가 구해지면, 이 바운딩 박스를 여러 개의 공간 셀로 분할한다. 각 공간 셀은 3차원 정수 인덱스가 할당된다. 객체를 구성하는 각 입자들은 어떤 셀에 속하는지가 계산되고, 같은 셀에 있는 입자들 사이에 충돌 회피 작업이 이루어진다.

그러나 이 방법은 셀의 수가 지나치게 많다. 이런 문제를 해결하기 위해 우리는 각 셀의 인덱스를 해쉬(hash)하여 $O(n)$ 개의 버켓(bucket)에 할당한다. 그림 1이 이러한 방법을 설명하고 있다. 어떤 정점 i는 그림에서 보는 바와 같이 하나의 셀에 포함되게 되고, 이 셀은 정수로 이루어진 3차원 좌표로 표현된다. 이 좌표값을 해쉬 함수의 입력으로 넘기면, 해쉬 함수가 특정한 해쉬값 k를 출력하고, 정점 i는 버켓 k에 저장된다.

이 방법은 각 입자들의 인접성을 확인하기 위해 필요한 공간 복잡도가 $O(n)$이며, 객체가 잘 흩어져 있는 경우에는 각 버켓에 $O(1)$ 개의 입자가 할당되므로 $O(n)$의 수행 속도를 보이게 된다.

이러한 충돌 회피 알고리즘을 수행할 때, 가장 큰 문제가 되는 부분은 매우 가까운 두 정점이 서로 다른 셀에 놓일 수가 있다는 것이다. 이러한 문제를 해결하기 위해, 옷감 객체가 차지하고 있는 공간을 분할하는 작업을 두 번 적용하였다. 즉, 한 번 공간을 분할하여 각 정점의 공간상 인덱스를 찾아 인덱스를 해쉬한 값을 얻은 뒤에, 공간 분할을 셀 크기의 반 정도로 x, y, z 축으로 각각 이동시킨 뒤에 다시 인덱스를 얻어 이를 또 다시 해쉬하여 버켓에 할당하는 것이다.

앞서 설명한 방법을 통해 우리는 충돌하는 두 정점을 효과적으로 찾을 수 있었다. 충돌하는 두 정점을 찾은 뒤에는 적절한 방법으로 이 두 정점의 충돌을 회피해야 한다.

본 논문에서는 충돌하는 두 정점을 감지한 후, 각각의 질점의 위치와 속도를 변경할 때, 충돌 임계치 θ_c를 사용한다. 충돌 임계치는 두 정점이 가질 수 있는 최소 거리라고 할 수 있다. 앞서 설명한 충돌 감지 방법에서 공간을 분할하는 셀의 크기를 이 충돌 임계치 크기로 사용하였다. 충돌하는 두 정점 i, j가 결정되면, 이 두 정점의 위치와 속도를 변경하기 위해 몇 가지 필요한 정보가 계산된다. 이 때 필요한 정보는 다음과 같다:

i와 j의 충돌 예상 지점 C_{ij}

충돌 지점의 속도 v_c

충돌 법선 벡터 N_c

충돌 지점 C_{ij}는 간단히 i, j 두 질점 위치의 중간 지점, 즉 $(\mathrm{x}_i + \mathrm{x}_j)/2$로 구했다. 충돌 지점의 속도 역시, 두 질점의 속도를 평균하여 $(\mathrm{v}_i + \mathrm{v}_j)/2$로 구하였다. 충돌 면의 충돌 법선 벡터는, 각 질점의 위치를 수정할 때, 면을 뚫고 들어가지 않도록 충돌 면에서 멀어지는 방향으로 수정할 수 있게 해 준다. 이 충돌 법선 벡터는 $(\mathrm{x}_i - \mathrm{x}_j)/|\mathrm{x}_i - \mathrm{x}_j|$를 사용하였다. 이 충돌 법선 벡터는 i의 위치를 수정하기 위한 것이며, j의 위치를 수정하기 위한 충돌 법선 벡터는 방향을 반대로 한 벡터가 된다. C_{ij}, v_c, N_c가 정해지면 위치의 수정이 이루어진다. 위치의 수정은 매우 간단한 작업으로 두 정점 i, j가 충돌 임계치 이내로 접근했을 때는 충돌 법선 방향으로 이동시켜 임계치 이상의 거리를 유지하도록 하는 것이다. 즉, x_i는 다음과 같이 수정된다.

$$\text{if} \quad |\mathrm{x}_{ji}| < \theta_c \quad then \quad \mathrm{x}_i = \mathrm{C}_{ij} + \theta_c \mathrm{N}_c \tag{8}$$

충돌하려는 정점의 위치를 강제로 수정함으로써, 현재 프레임에서의 충돌은 막을 수 있다. 하지만, 이렇게 위치만을 수정하면 충돌하는 두 객체의 속도가 그대로 유지되어 충돌 이후에도 계속해서 충돌하는 방향으로 움직이게 되어 이후 프레임에서 충돌을 회피할 수가 없다. 따라서 충돌 지점의 속도와 충돌하는 객체의 속도를 비교하여 충돌하는 객체가 충돌지점으로 다가가고 있는 경우에는 속도를 감속하여 충돌이 일어나지 않도록 해야만 한다. 이를 위해 충돌 객체 i와 충돌 지점의 상대 속도 v_{rel}를 다음과 같이 구한다.

$$\mathrm{v}_{rel} = \mathrm{v}_i - \mathrm{v}_c \tag{9}$$

이 상대 속도와 충돌 법선 벡터 N_c의 내적이 음수가 되면 충돌 객체 i가 충돌 지점에 다가가고 있다는 것을 알 수 있다. 따라서 이러한 경우에는 충돌 지점으로 다가가는 성분을 제거하는 작업을 수행한다. 이러한 작업은 다음과 같은 방법으로 이루어진다.

$$\mathrm{v}_i^{adjusted} = \mathrm{v}_i - (\mathrm{v}_{rel} \cdot \mathrm{N}_c)\mathrm{N}_c \tag{10}$$

이렇게 속도를 변경함으로써 자연스러운 충돌 회피 결과를 얻을 수 있다. 그림 2는 본 논문에서 제시하는 자체충돌 처리 기법을 이용하여 생성한 애니메이션의 결과이다.

그림 2. 자체충돌이 처리된 결과

V. 실험결과

본 논문의 기법은 일반적인 PC 환경에서 실시간 옷감 애니메이션을 구현하기 위해 연구되었다. 본 논문의 기법을 실험하기 위해 현재 일반적으로 사용되고 있는 Pentium-4 2.4 GHz 환경을 사용하여 테스트 하였다.

자체 충돌을 처리하여 생성한 실시간 애니메이션의 결과는 그림 3에 나타나 있다. 그림 3의 옷감은 자체 충돌을 처리하면서 바람에 날리는 옷감 애니메이션을

생성한 결과이다. 그림에서 볼 수 있는 바와 같이, 매우 사실적인 결과를 생성할 수 있었다. 이 애니메이션은 실시간에 생성되었고, 바람의 방향은 사용자가 임의로 조작할 수 있도록 하였다. 그림 3과 같은 애니메이션을 실시간에 생성하는 데에 본 논문의 기법은 매우 잘 동작하였다.

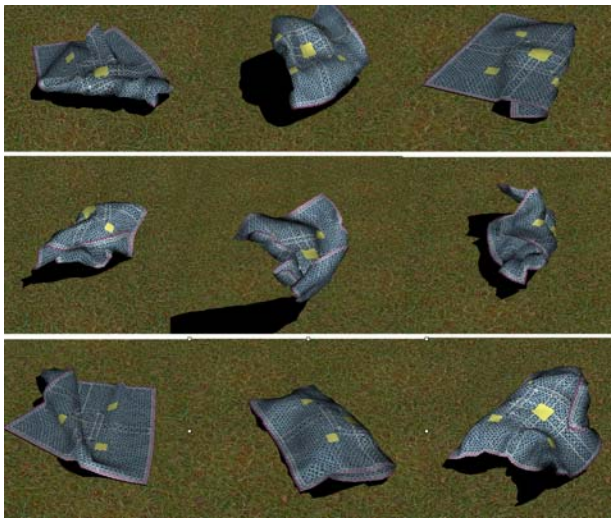

그림 2. 본 논문의 기법으로 생성된 실시간 애니메이션

VI. 결론

본 논문에서는 옷감 객체의 움직임을 가상현실 환경에서 실시간에 생성할 수 있는 효율적인 기법을 제안하였다. 이전에 제안된 여러 가지 실시간 기법들은 힘의 미분치를 지나치게 근사하여 안정성은 보장하지만 동작의 사실성이 크게 떨어지는 단점을 가지고 있었다. 본 논문에서 제안된 기법은 옷감 애니메이션의 사실성과 시간 효율성을 높이기 위해 기하적 기법을 사용하지 않고 수치적분의 해를 효과적으로 근사하고 있다. 따라서 본 논문에서 제안된 기법은 일반적인 옷감 메쉬 모델의 실시간 애니메이션을 쉽게 생성할 수 있게 해 준다. 또한, 본 연구에서 제안된 기법이 사용하는 힘의 미분치는 이전의 근사 기법과 달리 가능한 정확한 값을 갖도록 하여 움직임의 사실성을 더욱 높였다. 본 논문에서 제안된 기법을 이용하면, 사실적인 옷감 주름을 표현할 수 있는 복잡한 메쉬(mesh)의 자연스러운 움직임을 실시간 환경에서 얻을 수 있다.

옷감 애니메이션에서 애니메이션과 함께 계산 효율을 떨어뜨리는 중요한 요소는 충돌 문제이다. 옷감과 같이 변형 가능한 객체는 일반적인 충돌 문제와 함께 자체 충돌이라 불리는 특별한 충돌 문제를 가지는데, 이는 객체의 일부가 자신의 다른 부분에 충돌하는 문제이다. 이러한 자체 충돌을 해결하지 않으면, 애니메이션의 사실성을 얻을 수가 없는데, 이 자체 충돌을 해결하는 작업이 애니메이션 생성 기법의 시간효율을 심각하게 저하하게 된다. 본 논문에서는 이러한 자체 충돌 문제를 실시간 환경에서 효과적으로 처리할 수 있는 충돌 회피 기법을 제안하였다.

참고문헌

[1] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pp. 43-54, 1998.

[2] A. Vlachos, J. Peters, C. Boyd, and J. Mitchell. Curved PN triangles. *Symposium on Interactive 3D Graphics 2001*, pp. 159-166, 2001.

[3] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. *ACM Transactions on Graphics: Proceedings of SIGGRAPH 2002*. pp. 604-611. 2002.

[4] Frederic Cordier and Nadia Magnenat-Thalmann. Realtime animation of dressed virtual humans. *Proceedings of Eurographics 2002*, 2002.

[5] Mathieu Desbrun, Peter Schröder, and Alan Barr. Interactive animation of structured deformable objects. *Graphics Interface '99*, pp. 1-8, 1999.

[6] Bernhard Eberhardt, Andreas Weber, and Wolfgang Strasser. A fast, flexible particle-system model for cloth draping. *IEEE Computer Graphics & Applications*, 16(5):52-59, September 1996.

[7] Young-Min Kang, Jeong-Hyeon Choi, Hwan-Gue Cho, and Chan-Jong Park. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer Journal*, 17(3):147-157, 2001.

[8] M. Kass. An introduction to continuum dynamics for computer graphics. *In SIGGRAPH Course Note: Physically-based Modelling.* ACM SIGGRAPH, 1995.

[9] Masaki Oshita and Akifumi Makinouchi. Real-time cloth simulation with sparse particles and curved faces. *Proc. of Computer Animation 2001*, pp. 220-227, November 2001.

[10] Xavier Provot. Collision and self-collision handling in cloth model dedicated to design. *Computer Animation and Simulation '97*, pp. 177-190, September 1997.

[11] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):205-214, July 1987.

[12] Pascal Volino, Martin Courshesnes, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *Proceedings of SIGGRAPH 95*, pp. 137-144, August 1995.

[13] Pascal Volino and Nadia Magnenat Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum*, 13(3):155-166, 1994.