東京大学大学院新領域創成科学研究科
人間環境学専攻

修士論文

ボクセルベース衝突判定法を用いた物理シミュレーションに関する研究

2007年2月20日提出

指導教員　鈴木　克幸　助教授　印

学生証番号　５６８０７

牧野　哲也

目次

図目次

表目次

第1章 序論

1.1 研究背景

近年、CG アニメーションの分野で物理シミュレーションの需要が高まってきている。物理シミュレーションとは、現実世界の物理法則をコンピュータ内に取り入れて、さまざまな挙動や現象をコンピュータによる計算を行うことで表現しようというものである。物理シミュレーションを用いることで、人間が入力する部分を初期条件だけに抑えられるため、時間や労力の削減につながるだけでなく、アニメーション作成のための専門的な知識なども必要なくなることで誰でも簡単にアニメーションが作成できるという利点がある。

物理シミュレーションは現在多くの研究がなされており、市販ソフトウェアの中にも構造解析やクロスシミュレーション、流体シミュレーションなどのアニメーションツールが搭載されている。

しかし、需要の拡大および多様化に伴い、リアルタイムで計算できるだけでなく、より大きく複雑な問題を扱える物理シミュレーションが必要となってきている。複雑な挙動を表現するためにはアルゴリズムを複雑にする必要があるが、そのために計算効率が悪化したり、システムが不安定になったりするだけでなく、求めていた挙動が得られないなど精度の点でも多くの問題が残されている。

以上のことから、リアルタイム性と精度という二つの相反する条件を満たす物理シミュレーションのアルゴリズムが求められている。

1.2 研究目的

以上のような物理シミュレーションの現状を踏まえ、本研究では物理シミュレーションの中でも特に剛体運動シミュレーションを対象として、精度が高く効率的なシミュレーションを行うためのアルゴリズムの提案およびプログラムによる開発を目的とした。

剛体運動シミュレーションのアルゴリズムは、大きく分けて衝突判定と撃力計算という二つのフェイズから成り立っている。衝突判定においては久保田[1]によって提案されたボクセルベース法を採用するとともにその特徴について考察し、その手法に適した撃力計算手法を検証していくという手順で研究を進めた。

1.3 本論文の構成

本論文の第 2 章以降の構成は以下の通りである。

第 2 章では、剛体運動シミュレーションにおける既存手法について紹介し、既存手法の問題点を指摘する。

第 3 章では、剛体運動シミュレーションにおける衝突判定手法としてボクセルベース衝突判定法を提案し、その詳細と利点について述べる。

第 4 章では、剛体運動シミュレーションにおける撃力計算手法を選択するためにさまざまな手法について検証し、制約ベース法について詳細を述べる。

第 5 章では、制約ベース法における計算時間を小さくするための簡略化手法として見かけの衝突点を用いたシミュレーションを提案し、その詳細について述べる。

第 6 章では、ボクセルベース衝突判定法を用いた制約ベースシミュレーションを実装した結果を示し、撃力ベースシミュレーションや見かけの衝突点を用いたシミュレーションなどとの比較検証を行う。また、MATLAB 上で作成したアニメーション例および Cell 上でのアニメーション例を示す。

第 7 章では結論について述べる。

第2章 物理シミュレーションに関する既存研究

2.1 概要

本研究を始めるにあたり、物理シミュレーション、特に剛体運動シミュレーションに関する既存のさまざまな研究を調査した。まずは剛体運動シミュレーションの流れについてFig. 2-1に示す。

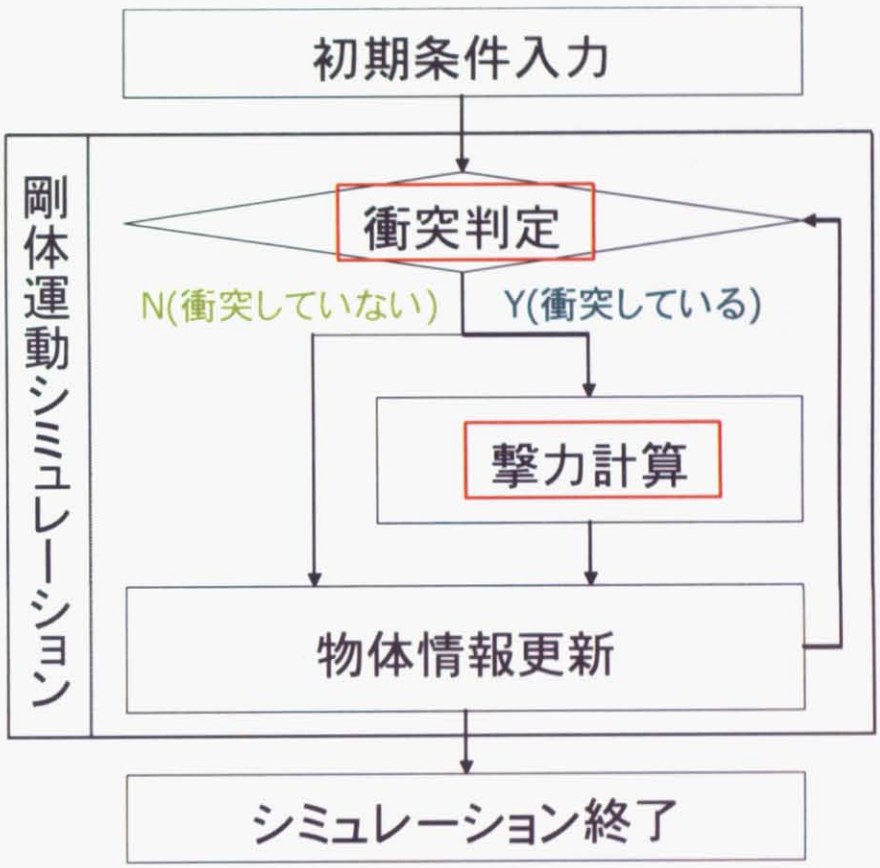

Fig. 2-1 剛体運動シミュレーションの流れ

物理シミュレーションに関する研究の主眼としては以下の二つが挙げられる。

- 物体間の衝突判定手法
- 衝突時の撃力計算手法

その他にも時間積分手法に関する研究などは存在するが、盛んに行われている研究は上の二つである。以下の節では、衝突判定手法、撃力計算手法に関する既存研究について紹介する。

2.2 衝突判定手法に関する既存研究

2.2.1 ポリゴンを用いる衝突判定手法

シミュレーションにかかる計算時間のうちの多くが物体同士の衝突判定に費やされるため、効率的な衝突判定手法の開発が必要である。また、モデルの形状をできるだけ損なわないようなモデリングをしないと、求める挙動が得られなくなってしまうため、正確に物体形状を反映できるような手法である必要もある。

衝突判定に関してはさまざまなアルゴリズムが提案されている[2][3][4][5][6]。一般に多面体同士の衝突判定手法では、ポリゴン対ポリゴンの衝突を考えるポリゴンベース衝突判定手法が多く用いられている。ポリゴンベース衝突判定法のメリットとしては、物体形状を正確にモデルに反映できるという点がある。モデルを作成するためには既存のソフトウェアを使用することが多いが、多くのソフトウェアは物体モデルをメッシュを用いて作成しているため、それをそのまま衝突判定用のポリゴンとして使用できるのである。

しかしながら、ポリゴンベース衝突判定手法では、点と面の判定や線と線の判定といったように、衝突の種類により衝突判定のアルゴリズムが異なるといった問題がある。また、その衝突の後の挙動のシミュレーションを行うには衝突の際に生じる力を評価する必要があるが、これも衝突判定の際と同様、衝突の種類により異なるため、アルゴリズムが非常に複雑となり、計算効率の低下を招く要因となる。また、手法の脆弱性から、しばしば物体が互いに貫通するような現象が生じ、これを防止するためには積分のタイムステップを調整する必要があるが、タイムステップに関しても明確な指針を示すことは困難である。

2.2.2 モデルを簡単な形状で近似する衝突判定手法

ポリゴンを用いる手法に対して、物体モデルを球や円柱などの簡単な形状の集合で近似する手法も提案されている。この手法では、衝突判定を近似した球や円柱で行うため、アルゴリズムを場合分けする必要がほとんどなく、また高速な判定が可能という利点がある。

この手法の問題としては、実際の物体に対して衝突判定に用いるモデルがどれほどの近似であるかという部分が大きい。高速に衝突判定を行いたいのならばできるだけ少ない球や円柱などの集合にしたいが、正確に衝突判定をしたいならば、できるだけ多くの球などで近似することで実際の物体形状に近付けなければならない。目的によってモデリングをし直さなければならないということが課題となる。

2.3 撃力計算手法に関する既存研究

2.3.1 ペナルティ法

撃力計算手法に関しても、さまざまなアルゴリズムが提案されている。大きく分けて、計算コストや挙動の精度という点が異なる。

ペナルティ法は、他の物体あるいは床への貫入量に応じて力を計算するという手法である。衝突点を i とすると、ペナルティ法における力の計算は、弾性係数 k、減衰係数 c を用いて以下の式で求められる。

$$\mathbf{f} = \sum_i \left(-kx_i - cv_i\right)\mathbf{n}_i \tag{2.1}$$

ここで、$\mathbf{n}$ は衝突点における撃力の方向を示す単位ベクトルである。

ペナルティ法は、非常に簡易なアルゴリズムであるため高速な計算が可能である。しかしながら、物体の貫入を許容しなければならないといった大きな欠点がある。物体の貫入を小さくするためには弾性係数を大きくしなければならないが、そのためにはタイムステップを十分に小さく取らなければならない。そのため結果的に計算時間は増大してしまうといった問題が生じてしまう。また、実際の反発係数などを用いないため、弾性係数や減衰係数といった剛体運動では見慣れない値の細かい調整を行わなければならない。

本研究ではペナルティ法の調査の一環として、弾性係数と減衰係数からの反発係数の計算式を求めた。

$$e = \exp\left(\frac{-c}{\sqrt{4mk - c^2}}\pi\right) \tag{2.2}$$

上記の式で反発係数を求めることができるが、ある反発係数と等しい弾性係数、減衰係数は上式をもとに調整する必要があり、タイムステップなどと合わせて値の試行錯誤が必要なアルゴリズムであるといえる。

2.3.2 撃力ベース法

撃力ベース法は、多点衝突を一転衝突の重ね合わせとして衝突時の力積を計算する手法である[7][8]。物体 i と物体 j がある衝突点で衝突した場合の、撃力ベースにおける力積は以下の式で表される。

$$\mathbf{p} = \frac{-(1+e)\mathbf{v}_{rel} \cdot \mathbf{n}_{ij}}{\frac{1}{m_i} + \frac{1}{m_j} + \mathbf{n}_{ij}\left\{\left(\mathbf{I}_i^{-1}\left(\mathbf{r}_i \times \mathbf{n}_{ij}\right)\right) \times \mathbf{r}_i + \left(\mathbf{I}_j^{-1}\left(\mathbf{r}_j \times \mathbf{n}_{ij}\right)\right) \times \mathbf{r}_j\right\}} \tag{2.3}$$

ここで、eは反発係数、$\mathbf{v}_{rel}$は物体iと物体jの相対速度ベクトル、$\mathbf{n}_{ij}$は法線方向単位ベクトル、mは質量、$\mathbf{I}$は慣性テンソル、$\mathbf{r}_i$は物体重心からの衝突点位置ベクトルである。床との衝突における撃力を求めたい場合は、$m_j = \infty$、$I_j^{-1} = 0$とすれば求めることができる。

衝突点が多点になった場合は上記の式の重ね合わせで力積を表現する。この手法もアルゴリズムとしては平易であるため、高速な計算が可能となる。また、貫入を考える必要が無く、実際の反発係数を用いているためパラメータの設定も容易であるという利点がある。

しかしながら、多点衝突を 1 点衝突の重ね合わせで表現するため、多点衝突の場合には実際の挙動と大きく異なってしまう可能性があるのが欠点である。このため、同じ物体をモデリングする場合でも、モデルの詳細度が違うと全く異なった挙動になってしまうこともあるなどの現象が生じてしまう。

2.3.3 制約ベース法

制約ベース法は、衝突点の数と等しい制約条件を作成し、連立不等式を解くことで力積を求めるという手法で、Baraff[9][10]によって考案された。

質点において衝突前の法線方向相対速度 v と衝突後の法線方向相対速度 v'の関係式は、反発係数 e を用いて以下のように表される[11]。

$$v' = -ev \tag{2.4}$$

Baraff はこれを剛体の多点衝突に拡張し、衝突点$i = 1 \sim n$に対して、

$$v_i' \geq -ev_i \tag{2.5}$$

であることを必要条件とした。式が不等式になっているのは、剛体の多点衝突では質点の衝突と異なり、ある点での衝突が他の点の法線方向相対速度に変化を及ぼすため、この影響を考慮したという理由による。例として、Fig. 2-2のような状況では、衝突前の衝突点 1 の法線方向相対速度は 0 であるが、衝突点 2 の衝突時撃力の影響により、衝突後の衝突点 1 の法線方向相対速度は正となる。

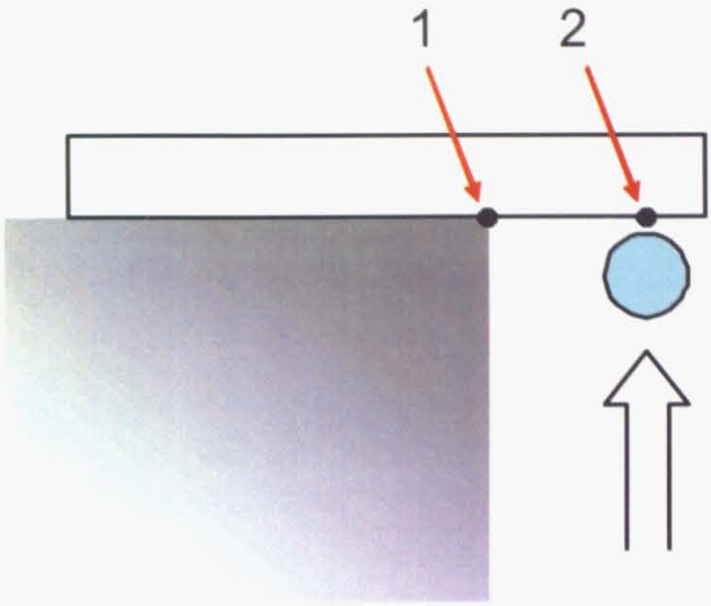

Fig. 2-2 他の衝突点の影響により $v_1 > -ev_1$ となる例

また、衝突時の撃力は物体を互いに引き離す方向にしか働かないので、衝突点 i においては、

$$p_i \geq 0 \tag{2.6}$$

が成立する。さらに、$v_i > -ev_i$ が成立する場合には、衝突点 i 以外での衝突時撃力によって衝突点 i では物体が引き離されると考えられるため、衝突点 i では撃力は働かない(Fig. 2-2は、衝突点 1 において $v_1 > -ev_1$ となる例)。

以上の条件を考慮して、ある物体に対して k 個の衝突点があるとすると、制約ベース法では以下の 4 つの式が立てられる。

$$\begin{aligned} \mathbf{V'} &= \mathbf{Ap} + \mathbf{V} \\ \mathbf{V'} &\geq -e\mathbf{V} \\ \mathbf{p} &\geq \mathbf{0} \\ p_i\left(v' + ev\right)_i &= 0 \left(i = 1 \sim k\right) \end{aligned} \tag{2.7}$$

ここで、**V** は衝突前の法線方向相対速度を並べたベクトル、**V'**は衝突後の法線方向相対速度を並べたベクトル、**p** は法線方向力積の大きさを並べたベクトル、**A** は衝突点の位置ベクトルや法線方向単位ベクトルあるいは物体の質量などから計算される係数行列である。この問題は線形相補性問題と呼ばれ、制約条件式を満たす **p** を求めることで衝突時の撃力の大きさを求めることになる。

制約ベース法は、このように多点衝突においても一度に解を求めるので、正確な挙動を実現できる。反面、制約式を立て、いわゆる線形相補性問題を解かなければならないため、計算時間が大きくなってしまうという欠点がある。

2.3.4 SEM 法

SEM 法とは、System Energy Minimize Method の略であり、系全体のエネルギーが最小化されるように接触力を求めるという手法である[12]。ここでは詳細は省き、例として系の中に物体がひとつだけの場合を考えてみる。このとき時刻 t における系のエネルギー E_t は、

$$E_t = \frac{1}{2} m \mathbf{v}_t^{\mathrm{T}} \mathbf{v}_t + \frac{1}{2} \boldsymbol{\omega}_t^{\mathrm{T}} \mathbf{I}_t \boldsymbol{\omega}_t - m \mathbf{g}^{\mathrm{T}} \mathbf{x}_t \tag{2.8}$$

で表される。同様に時刻 $t+\Delta t$ における系のエネルギー $E_{t+\Delta t}$ を衝突時の接触力 $\mathbf{f}$ の関数として表し、

$$\Delta E = E_{t+\Delta t} - E_t \rightarrow \min \tag{2.9}$$

を満たす接触力 $\mathbf{f}$ を求めることで力が計算される。

SEM 法は、系全体のエネルギーを考えるため、制約ベース法と同様に正確な挙動が得られると考えられる。しかしながら、計算時間はやはり大きくなってしまうことになる。また、式の中に反発係数や摩擦係数、さらに床や壁面のエネルギーが入っていないため、場合によっては正確な挙動が得られないこともありうるのが問題となる。

2.3.5 ベクトル場を用いる手法

ベクトル場を用いる手法は、物体の周囲に斥力を与えるベクトル場を定義し、自動的に物体の衝突回避を行う手法である[13]。2 つの物体が接近した場合には、物体周囲の斥力ベクトル場が相互に影響することで、物体が引き離される方向に力を受けることになる。

この手法は、あらかじめ物体に対して斥力ベクトル場を定義してしまえば高速な力の計算が可能であるが、斥力ベクトル場自体が現実とはかけ離れたものであるため、実際の挙動と大きく異なってしまうことが多々ある。また、物体同士の貫入の可能性もあるため、適切な斥力ベクトル場のパラメータを設定しなければならず、試行錯誤が必要な手法といえる。

2.3.6 既存手法の問題点

本章では、さまざまな既存の衝突判定手法および撃力計算手法について紹介してきたが、剛体運動シミュレーションを行う場合には衝突判定手法に適した撃力計算手法が必要となる。どの手法についても一長一短があるため、うまく長所を生かしたり、欠点を補ったりという相互の連携が求められる。

しかしながら、衝突判定手法と撃力計算手法の適切な組み合わせは試行錯誤が必要であり、衝突判定手法の特徴を正確に把握することが肝要である。

第3章 ボクセルベース衝突判定法

3.1 概要

剛体運動シミュレーションにおいては、衝突判定手法と撃力計算手法のウェイトが大きいことは前章で述べた。そこで本章では、前章で示したさまざまな課題を解決する衝突判定手法として、ボクセルベース衝突判定法を提案する。Dingliana and O'Sullivan の考え方をもとに、久保田ら[1]が拡張した手法である。

この手法は、前処理として物体を段階的にボクセル分割することから始まる。衝突判定の際も、大きなボクセルの外接球を bounding sphere として、段階的に小さなボクセルへと衝突判定のレベルを上げていく。そして最も小さなボクセルで衝突が確認されると撃力計算のフェイズへと移行する、という流れである。

以降では、ボクセルベース法の詳細とその利点について説明していく。

3.2 ボクセルベース衝突判定法の詳細

3.2.1 物体のボクセル分割

ボクセルベース衝突判定法では、シミュレーションを開始する前に物体のボクセルモデルデータを作成する必要がある。ボクセルデータ作成に用いる物体は、二次元の場合は三角形の集合、三次元の場合は四面体メッシュの集合で構成されているとする。その形状をFig. 3-1に示すように、形状全体を内包する立方体ボクセル(二次元では正方形)に分割し、物体が存在するボクセルについてのみ、その外接球をbounding sphere として衝突判定に利用する。

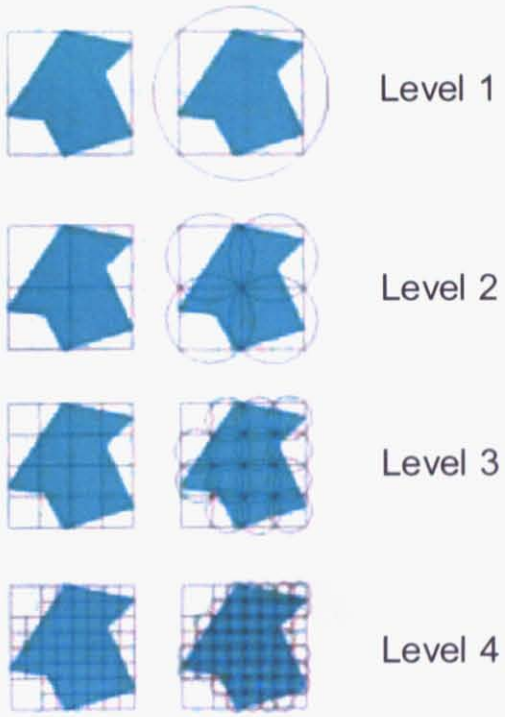

Fig. 3-1 二次元における物体のボクセル分割

Fig. 3-1を見ても明らかなように、、ボクセル分割は非凸形状の物体に対しても容易に行える。また、物体を規則正しく並んだ球の集合として表現することで、非凸形状物体同士の衝突判定の問題点を克服することが可能である。

実際の衝突は必ず表面で生じるため、物体表面に対応する球以外は利用しない。形状のボクセル分割のレベルを変更することにより、形状の粗い近似からより正確な近似まで様々なレベルでの表現が可能である。全体を 1 つのボクセルで表現した場合をレベル 1 とし、ボクセルの一辺の寸法を半分にするにつれてレベルを増やしていくこととする。Fig. 3-2に三次元の場合のボクセル分割と形状表現の例を示す。当然、衝突判定の精度を上げるために細かいボクセル分割を行えば、衝突判定に要

する計算時間が増加するため、要求精度に応じた適当なボクセル分割のレベル設定が必要になる。言い方を変えれば、レベルを変えることによる精度と計算時間の調整は従来手法よりはるかに容易に行える。

また、レベルが下のボクセル外接球は、レベルが上のボクセル外接球を完全に内包しているため、衝突判定のアルゴリズムも上から段階的に行っていけばよい、という利点もある。

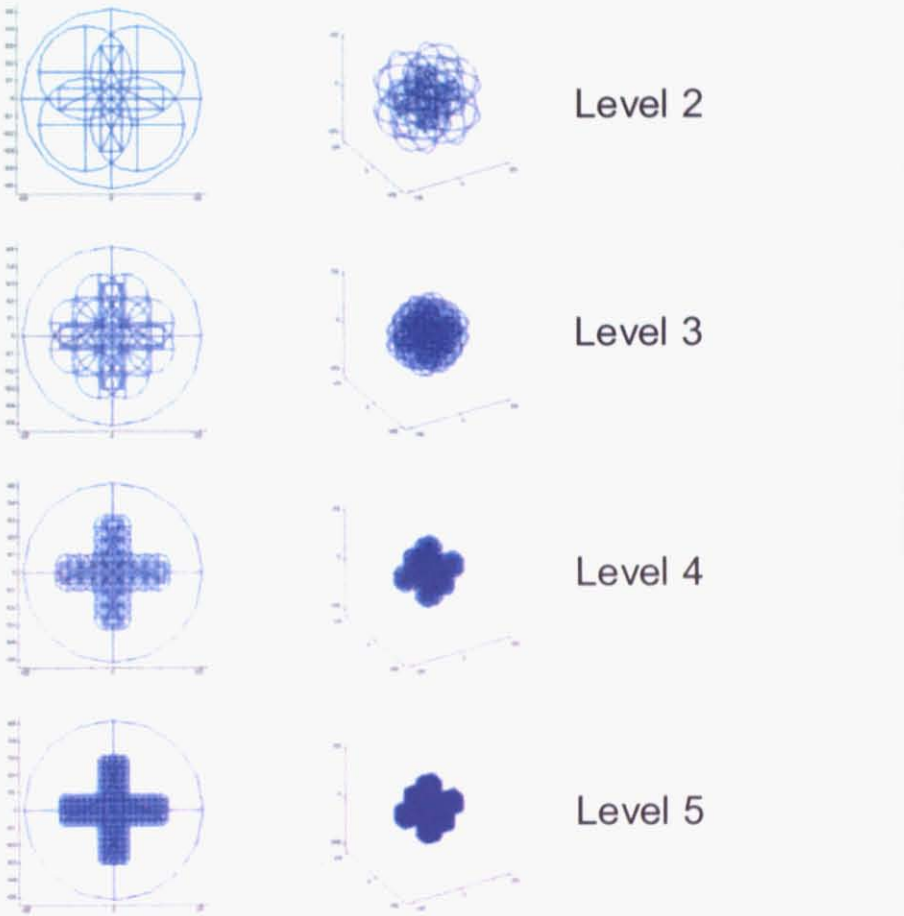

Fig. 3-2 三次元における物体のボクセル分割

3.2.2 ボクセルデータ作成のための in-out 判定

物体のボクセルデータを作成する際に問題となるのは、そのボクセル内に物体が存在しているかどうかの判定である。一般に in-out 判定と呼ばれ、さまざまな手法が提案されているが、本研究ではFig. 3-3のようなアルゴリズムで in-out 判定を行った。

```
In-Out detection of voxel centroid i
for every tetrahedral mesh
{
   for every surface triangle of the mesh
   {
      define a as a vector from i to a center of the triangle.
      define b as a normal outward vector of triangle.

      if(a・b>=0)
      {
         Return 0.
      }
      else
      {
         Return 1.
         break;
      }
    }

   if(Return is 0)
   {
      i is inside the object.
      break;
   }
}
```

Fig. 3-3 in-out 判定のアルゴリズム

要約すると、あるボクセル重心 i とある四面体メッシュ j に対して、ボクセル重心 i から j のある三角形メッシュ k の重心へのベクトルを **a** とする。また、k の外向き法線ベクトルを **b** とする。このとき、全ての三角形メッシュに対して **a** と **b** の内積がゼロ以上となるような四面体メッシュが存在すれば i は物体の内部にあると判定される、というものである。

実際にボクセルデータを作成する際は、最も大きなレベルのボクセルデータについては上に示したアルゴリズムを用いて作成するが、小さなレベルのボクセルデータに対しては別の手段を用いて作成する。Fig. 3-4に二次元での例を示す。

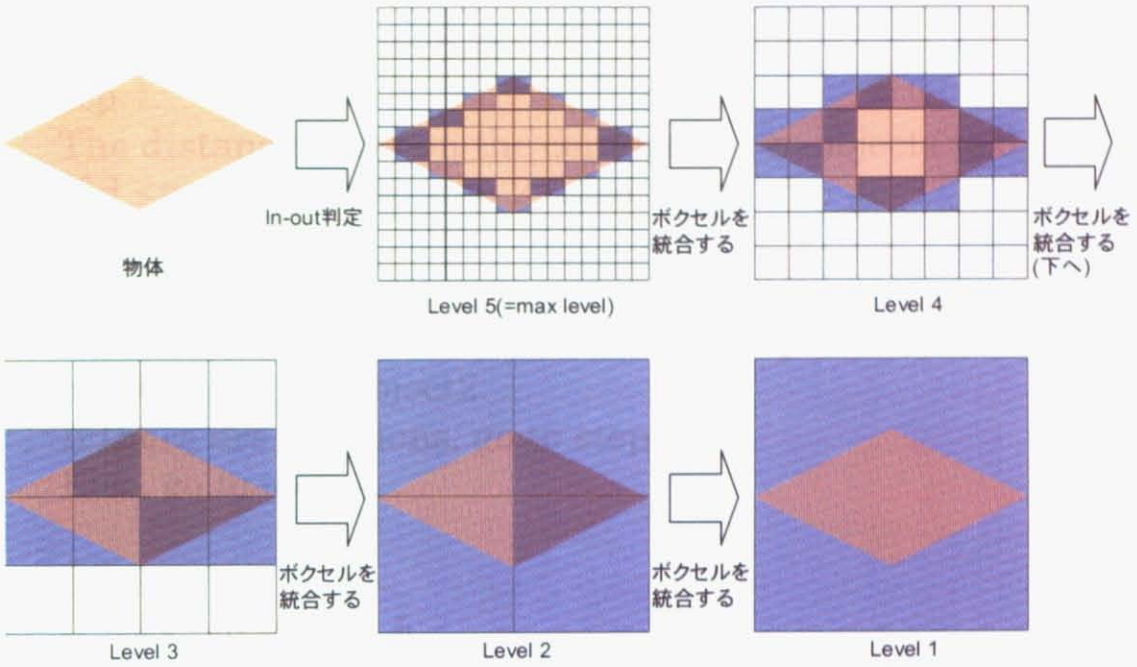

Fig. 3-4 物体のボクセルデータ作成手順

まず最大レベルのボクセルデータについて物体表面ではないと判定された部分を除外し、そのデータを 8 個 1 組(二次元では 4 個 1 組)としてひとつ小さいレベルのボクセルデータを段階的に作成していく。

3.2.3 衝突判定の流れ

衝突判定は、作成した段階的なボクセルデータを用いて行うが、どのレベルまで詳細化するかは自由に設定することが可能である。ボクセルレベル k における衝突判定の流れは以下のようになる。

```
Level k collision detection
 Step 1.
  The distance between the centers of two objects is d.
  If d <= r1+r2, go to step 2. Else, go to step 4.

 Step 2.
  Check collisions between the Level N spheres
  of Object1 and Object2.
  If there are collisions, go to step 3.
  Else, go to step 4.

 Step 3.
  If N=k, go to step 5.
  Else, divide the colliding spheres into Level N+1 spheres.
  And, N=N+1.

 Step 4.
  Collisions aren't detected. Return 0.

 Step 5.
  Collisions are detected. Return 1.
```

Fig. 3-5 ボクセルベース衝突判定の流れ

まず、レベル 1 の外接球同士の距離から衝突の可能性を調べ、衝突の可能性がある場合はレベル 2 の外接球同士の衝突判定を行う。そして、衝突をしていると判定されたレベル 2 のボクセルのみをレベル 3 に分割し、レベル 3 の外接球同士の衝突判定を行う。衝突をしていると判定されたレベル 3 のボクセルのみをレベル 4 に分割し、レベル 4 の衝突判定を行う。以上の判定を繰り返し、あらかじめ設定した衝突判定に用いる最大レベルの外接球においても衝突があると判明した場合に物体同士の衝突が起こったと判定する。

始めから細かい部分同士の衝突判定を行わず、判定に用いる球の大きさを段階的に小さくしていくことで、衝突に関係ない部分の余分な計算を省くことができるため、衝突判定の効率をあげることができる。

Fig. 3-6 段階的な衝突判定の例

3.3 ボクセルベース衝突判定法の利点

ボクセルベース衝突判定法を説明する途中でもいくつかの利点を挙げてきたが、ここで改めてボクセルベース法の利点についてまとめてみる。

- 非凸形状物体などに対しても問題なくシミュレーションが行える
- 球で衝突判定を行うので、アルゴリズムが平易である
- 要求する精度や計算時間に対しての調整が容易である
- 衝突部分だけを段階的に分割していくので、効率がよい
- ボクセルデータの容量が形状の複雑さに依存しない

これらの利点は、久保田[1]によって検証が行われており、ポリゴンを用いる衝突判定手法に対してその優位性が確認されている。剛体運動シミュレーションにおけるボクセルベース衝突判定法は、久保田の他、杉本[14]の研究にも用いられている。

第4章 制約ベース撃力計算法

4.1 概要

前章ではボクセルベース衝突判定法の詳細とその利点について示した。本章では、ボクセルベース法に適した撃力計算手法はどのようなものであるかを検証する。

第一段階として、ボクセルベース法の性質について検証してみた。その後、その性質に適した撃力計算手法を探していくという手順をとったが、検証する撃力計算手法は、既存研究で紹介した各手法である。特に現実的な挙動が期待できる撃力ベース法、制約ベース法については実際にプログラムを作成し、特に衝突時の挙動に対して検証を行った。

本研究のシミュレーション環境について以下に示す。

- OS: Microsoft Windows XP Professional SP2
- CPU: Intel Pentium 4 3.20GHz
- メモリ: 512MB
- プログラミング言語: MATLAB 7.01(MathWork 社)

4.2 撃力計算手法の検証・考察

4.2.1 ボクセルベース法の性質についての検証

ボクセルベース法の利点については前章で紹介したが、撃力計算手法の選択をする前に、その性質について利点だけではない性質について検証を行った。

その結果、ボクセルベース法の性質として判明したのは以下の 3 点である。

1. 計算時間は形状の複雑さにほとんど依存しない
2. 計算時間はボクセルレベルの大きさに依存する
3. 多点衝突になることが多い

1.と 2.はボクセルベース法のアルゴリズムを検証することで明らかとなる。どのような形状の物体も、同じレベルのボクセルデータに変換するとデータ容量は等しくなり、したがってメッシュデータで衝突判定を行う場合はポリゴン数に応じて衝突判定を行わなければならないが、ボクセルデータでの衝突判定においてはレベルが同じであれば探索に要する時間はそれほど変化しない。

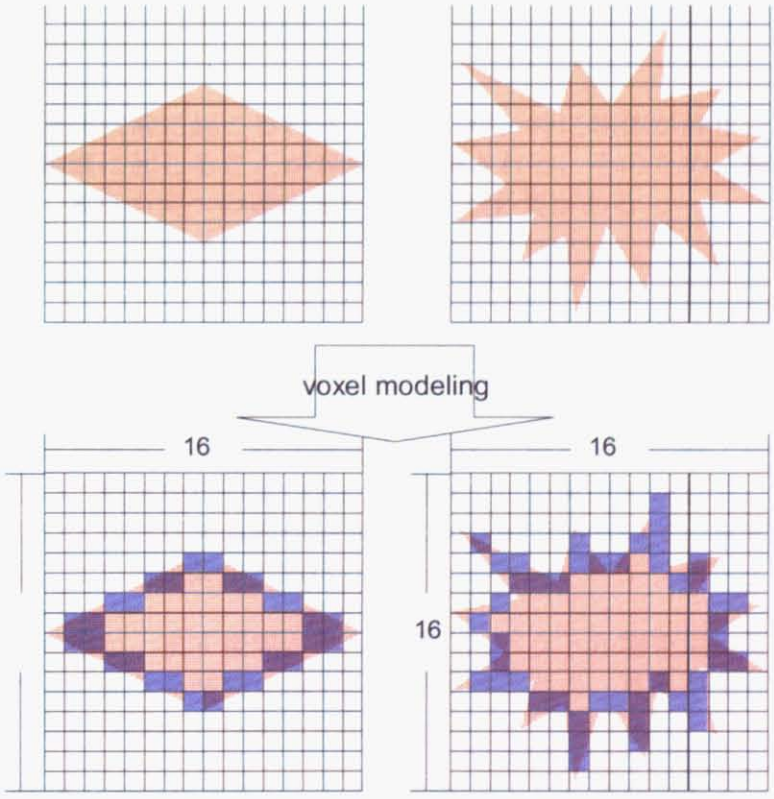

Fig. 4-1 単純な形状と複雑な形状のボクセルデータ化の例

二次元での例としてFig. 4-1を取り上げる。赤い部分が元の物体形状、青い部分がボクセルデータとして物体が存在する部分(ただし、表面のみ)である。ボクセルレベル 5 では単純な形状でも複雑な形状でも 16×16(三次元では 16×16×16)のボクセルデータを持つことになる。これはレベル 1～4 でも同様である。

撃力計算手法を選択する上では 3.が大きな意味を持つ。ボクセルベース法を用いた衝突判定では、ポリゴンの表面となる平面も球の集合として扱うため、ポリゴンにおいて平面同士の衝突や、非凸形状の衝突のような場合では、ボクセルレベルにも依存はするが、複数のボクセル外接球が衝突と判定されることになる。Fig. 4-2は立方体の平面が床と衝突する例であるが、ポリゴンでは平面と平面の衝突という 1 つの衝突と判定されるが、ボクセルベース法では、レベル 3 だと 4×4 のボクセル外接球が、レベル 4 だと 8×8 のボクセル外接球が、レベル 5 だと 16×16 のボクセル外接球が衝突と判定されることになるのである。

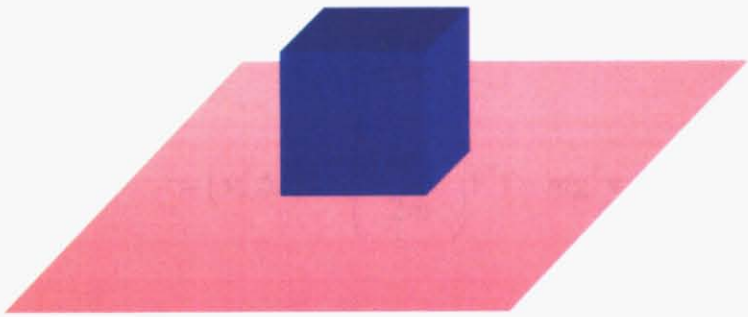

Fig. 4-2 複数のボクセル外接球で衝突と判定される例

もちろん多点衝突は常に起こるわけではないが、特に物体が静止に向けて収束している状況では頻繁に起こることが容易に推測される。そのため、撃力計算手法を選択する際にも、多点衝突への適応度を検証する必要がある。

4.2.2 各撃力計算手法の考察

4.2.1で示したボクセルベース法の性質を考慮し、既存研究の章で紹介した各種撃力計算手法について検証を行っていく。本研究では計算効率が良く、精度の高いシミュレーションのアルゴリズムを作成するのが目的であるので、まずは現実的な挙動を実現できることが必要条件である。

その側面から考察し、まず現実とはかけ離れたベクトル場を用いる手法は、現実的な挙動とは大きく異なる挙動を示すことが多いことと、貫入が生じないことが保障されていないという理由から除外した。

また、SEM 法は、系のエネルギーを最小化するという手法である。簡単のため二次元における質点の衝突で検証を行った。

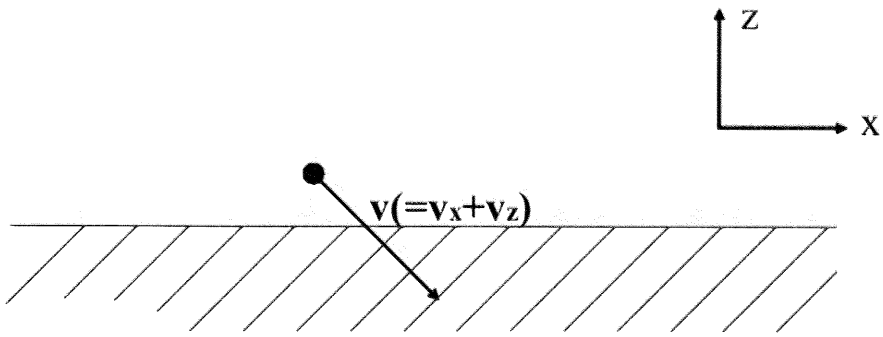

Fig. 4-3 二次元における質点の衝突例

Fig. 4-3はその一例である。SEM 法では、エネルギーEを撃力 **f** の関数で表すことから始まる。

$$\begin{aligned}\Delta E &= E_{t+\Delta t} - E_t \\ &= \left(\mathbf{v}_t^T \Delta t\right)\mathbf{f} + \left(\frac{(\Delta t)^2}{2m}\right)\mathbf{f}^{\mathrm{T}}\mathbf{f} - m\mathbf{g}^T\mathbf{v}_{\mathrm{t}}\Delta t\end{aligned} \tag{4.1}$$

ここで、

$$\mathbf{f} = -s_1\mathbf{q} - s_2\mathbf{d} \tag{4.2}$$

とおく。**q** は法線方向単位ベクトル、**d** は接線方向単位ベクトルを表す。SEM 法はΔEが最小となるような **f** を求める。つまり、

$$\frac{\partial \Delta E}{\partial s_i} = 0\left(i = 1,2\right) \tag{4.3}$$

となるようなs_iを求めることになるが、式(4.2)を式(4.1)に代入して式(4.3)を解くと、

$$\mathbf{f} = -\left(\frac{mv_z}{\Delta t}\right)\mathbf{q} - \left(\frac{mv_x}{\Delta t}\right)\mathbf{d} \tag{4.4}$$

と求められ、これを時間積分の式に代入すると、

$$\mathbf{v}_{\mathbf{t}+\Delta\mathbf{t}} = \mathbf{v}_{\mathbf{t}} + \frac{\mathbf{f}\Delta t}{m} = 0 \tag{4.5}$$

となり、どのような速度で床と衝突しても、床に張り付いて静止してしまうことになる。これは、質点がひとつである系において、エネルギーは質点が床上で静止しているときに最小になることを考えると明らかである。

また、ペナルティ法については、計算手法自体が物体同士の貫入を許容しなければならない手法であるため、これについても本研究の目的とは合致しないと判断した。

以上、ベクトル場を用いる手法と SEM 法、ペナルティ法については本研究の「正確で効率のよい剛体運動シミュレーションの実現」という目的に合致しないため、実装段階で除外することとした。

4.2.3 各撃力計算手法を用いたシミュレーションの検証

撃力ベース法、制約ベース法については、実際にプログラムを作成してシミュレーションを行うことで検証を行った。

シミュレーションは、解析解の算出が可能であるように、一物体の床への落下という簡易な例で行った(Fig. 4-4)。

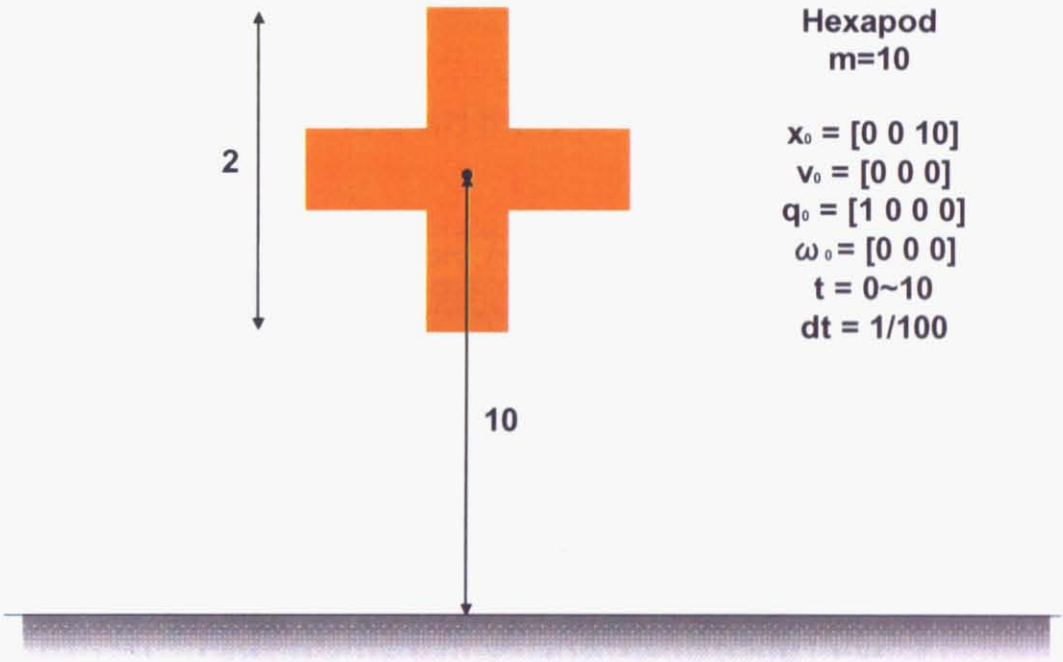

Fig. 4-4 検証に用いたシミュレーションの初期条件

以下は解析解とボクセルベース法の各レベルに各撃力計算手法を用いたシミュレーションの精度を検証したものである(Fig. 4-5、Fig. 4-6)。

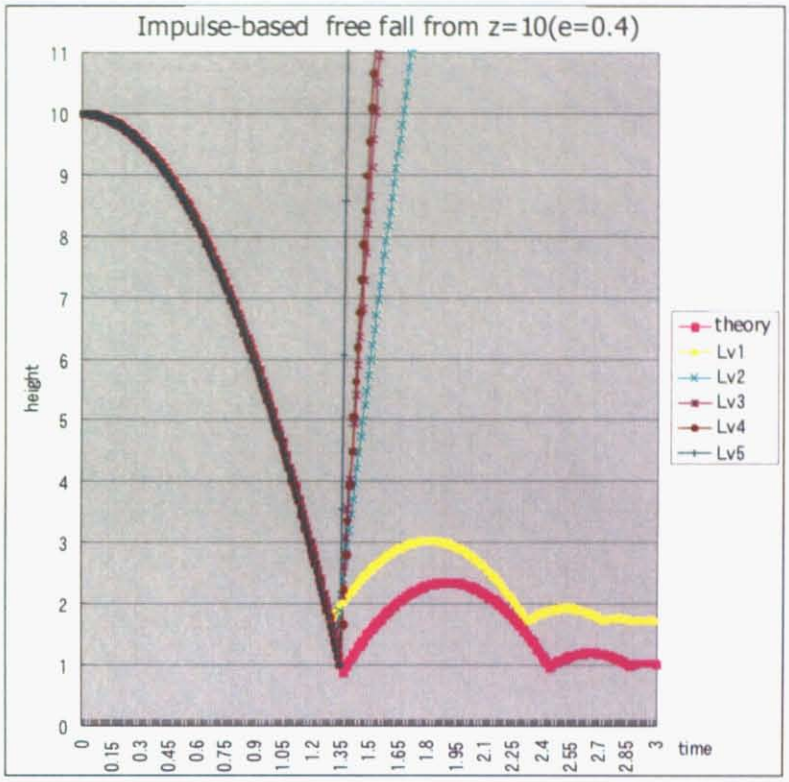

Fig. 4-5 各ボクセルレベルにおける撃力ベース法の自由落下シミュレーション

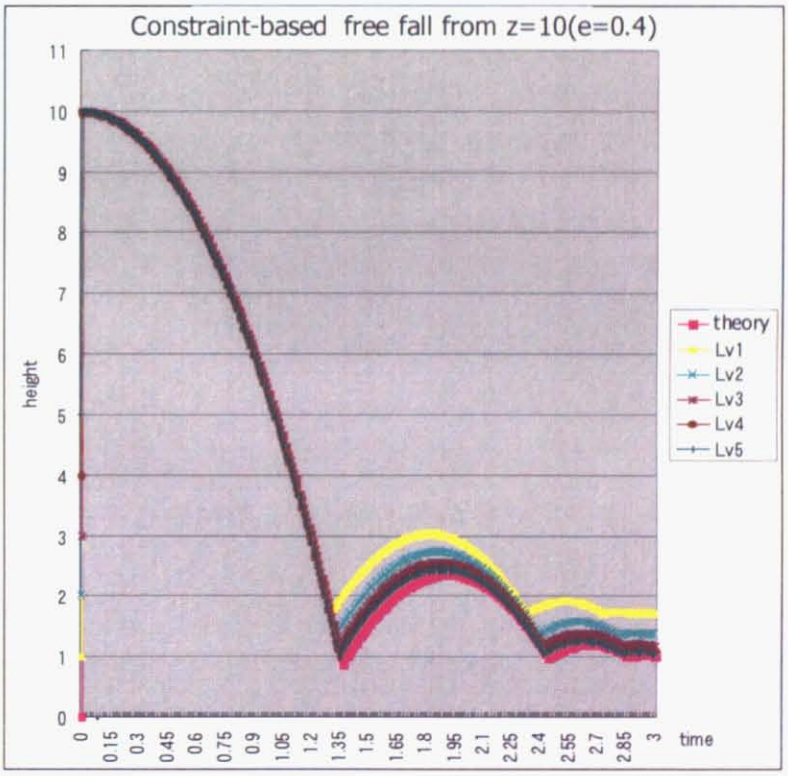

Fig. 4-6 各ボクセルレベルにおける制約ベース法の自由落下シミュレーション

各図を見ると、制約ベース法ではボクセルレベルが上がるごとに解析解に近づいていくのに対して、撃力ベース法では、ボクセルレベルが 2 以上、つまり複数のボクセル外接球が同時に衝突していると判定されるような場合は挙動が著しく異なってしまう。

これは、撃力ベース法では、複数点での衝突を 1 点衝突の重ね合わせで表現するのが原因である。同一法線方向に撃力を受ける点が複数存在すると、点の数が増えるほどその法線方向に撃力が足し合わさってしまうからである(Fig. 4-7)。

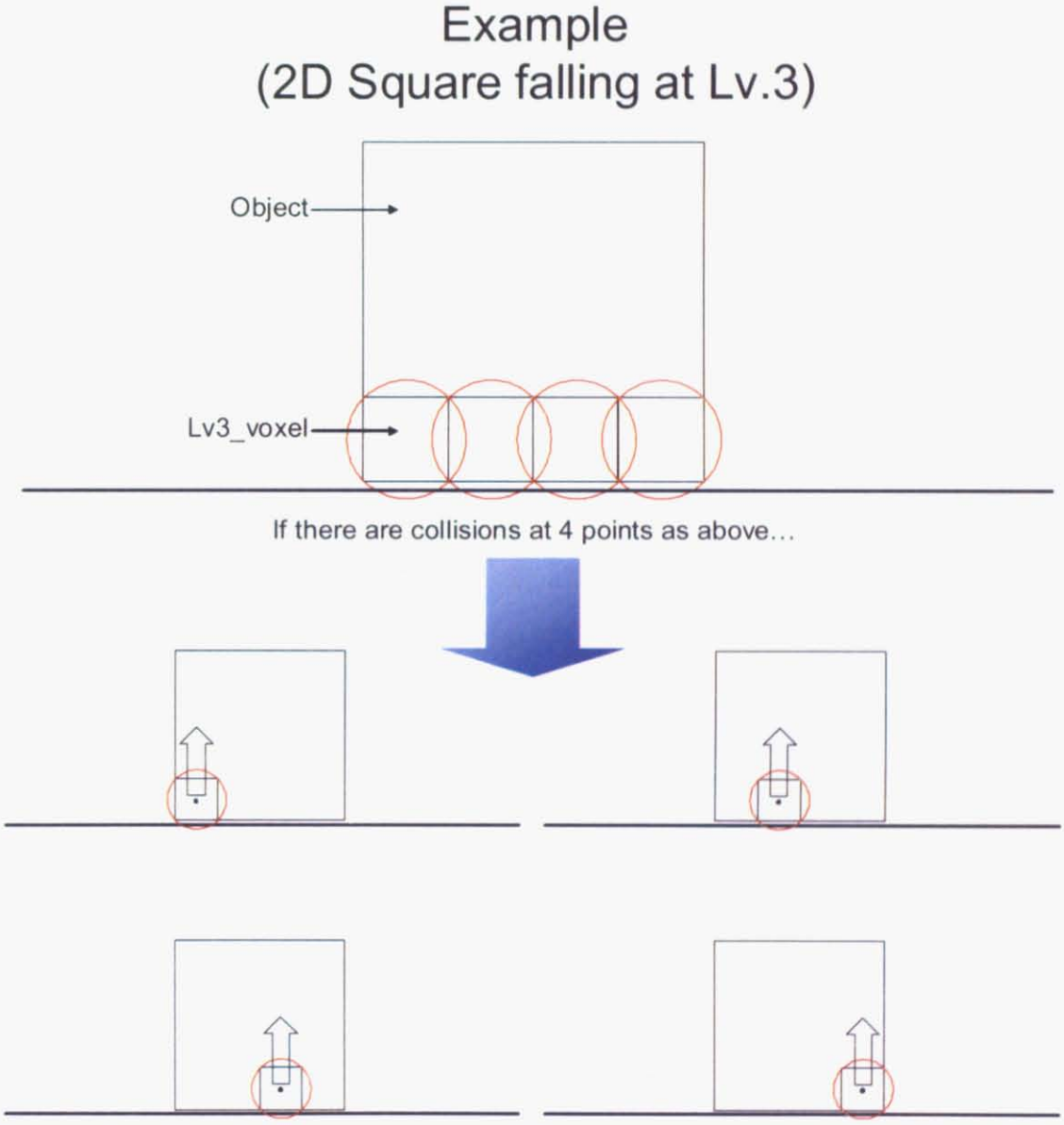

Fig. 4-7 撃力ベース法の問題点

一般的に、物理シミュレーションにおいて撃力ベース法を用いたアルゴリズムでは、複数点の衝突が生じた際は、各撃力を衝突点数で割るという処理を行っている。しかしこの処理は、物理的な意味が存在しないため、複数物体同士の衝突など挙動が複雑になると精度が落ちてしまう可能性がある。

そこで本研究では、複数点の衝突においても現実的なシミュレーションが可能な制約ベース法を用いて剛体運動シミュレーションを作成することを主目的とし、撃力ベース法を比較対象として、精度や計算時間などの側面から検証を行うこととした。

4.3 制約ベース法のアルゴリズム

4.3.1 制約ベース法の概要

第2章でも触れたが、制約ベース法のアルゴリズムについてさらに詳細に説明する。まず、Fig. 4-8のように物体Aと物体Bが1点で衝突している場合を考える。

物体A		物体B	
質量	: m_A	質量	: m_B
慣性テンソル:	$\mathbf{I}_A$	慣性テンソル:	$\mathbf{I}_B$
並進速度	: $\mathbf{v}_A$	並進速度	: $\mathbf{v}_B$
回転速度	: $\boldsymbol{\omega}_A$	回転速度	: $\boldsymbol{\omega}_B$

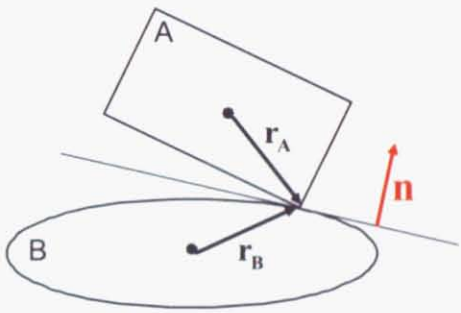

Fig. 4-8 二つの物体の1点衝突

このとき衝突前の物体A上の衝突点の速度 $\mathbf{v}_{r_A}$ は、

$$\mathbf{v}_{r_A} = \mathbf{v}_A + (\boldsymbol{\omega}_A \times \mathbf{r}_A) \tag{4.6}$$

同様に、衝突前の物体A上の衝突点の速度 $\mathbf{v}_{r_A}'$ は、

$$\mathbf{v}_{r_A}' = \mathbf{v}_A' + (\boldsymbol{\omega}_A' \times \mathbf{r}_A) \tag{4.7}$$

と表現できる。ここで衝突後の並進速度および回転速度は、衝突時の撃力 $p\mathbf{n}$ を用いて、

$$\mathbf{v}_A' = \mathbf{v}_A + \frac{p\mathbf{n}}{m_A} \tag{4.8}$$

$$\boldsymbol{\omega}_A' = \boldsymbol{\omega}_A + \mathbf{I}_A^{-1}\left(\mathbf{r}_A \times p\mathbf{n}\right) \tag{4.9}$$

と書くことができる。以上の式より衝突後の速度 $\mathbf{v}_{r_A}$ を撃力の大きさ p の関数として以下のように表すことができる。

$$\begin{aligned}\mathbf{v}_{r_A}' &= \left(\frac{\mathbf{n}}{m_A} + \left(\mathbf{I}_A^{-1}\left(\mathbf{r}_A \times \mathbf{n}\right)\right) \times \mathbf{r}_A\right) p + \mathbf{v}_{r_A} \\ &= \mathbf{c}_A p + \mathbf{v}_{r_A}\end{aligned} \tag{4.10}$$

ただし、$\mathbf{c}_A$ は既知のパラメータからなるベクトルである。

以上の計算を物体 B についても同様に行うと、

$$\mathbf{v}_{r_B}' = \mathbf{c}_B p + \mathbf{v}_{r_B} \tag{4.11}$$

と表せる。

ここで、衝突点における物体 A と物体 B の相対速度を v とおくと、

$$v = \mathbf{n} \cdot \left(\mathbf{v}_A - \mathbf{v}_B\right) \tag{4.12}$$

となるので、衝突後の相対速度 v' は、

$$\begin{aligned}v' &= \mathbf{n} \cdot \left(\mathbf{v}_A' - \mathbf{v}_B'\right) \\ &= \mathbf{n} \cdot \left(\mathbf{c}_A - \mathbf{c}_B\right) p + \mathbf{n} \cdot \left(\mathbf{v}_A - \mathbf{v}_B\right) \\ &= cp + v\end{aligned} \tag{4.13}$$

と撃力 p の関数で表すことができる。c は物体 A、B の既知パラメータからなる定数である。

以上 2 物体の 1 点衝突における撃力 p と衝突後の相対速度 v' との関係について示してきたが、これを n 点の多点衝突に拡張して、

$$v_i' = \sum_{j=1}^{n} a_{ij} p_j + v_i \tag{4.14}$$

と表すことができ、さらにベクトルと行列の式に書き直せば、

$$\mathbf{V}' = \mathbf{A}\mathbf{p} + \mathbf{V} \tag{4.15}$$

となる。この場合の撃力 $\mathbf{p}$ を求めるために、Baraff は、「衝突点 i における衝突後の法線方向相対速度 v_i' は、$v_i' \geq -ev_i$ でなければならない」、「衝突点 i における撃力

p_iは、物体を互いに引き離す方向にしか働かない」、「$v_i' > -ev_i$が成立する衝突点 iでは他の衝突点における撃力によって不等号が成立しているので、撃力$p_i = 0$となる」という制約条件を定めて、

$$\begin{aligned} \mathbf{V}' &= \mathbf{A}\mathbf{p} + \mathbf{V} \\ \mathbf{V}' &\geq -e\mathbf{V} \\ \mathbf{p} &\geq \mathbf{0} \\ p_i\left(v' + ev\right)_i &= 0 \left(i = 1 \sim k\right) \end{aligned} \tag{4.16}$$

の4つの式から撃力 **p** を求めるという手法を構築した。

この問題は線形相補性問題と呼ばれ、本研究ではその解法として二次計画問題への変換および二次計画法を採用した。

具体的には、式(4.16)を次のように変換する。

$$\mathbf{p}^{\mathrm{T}}\mathbf{V}' = \mathbf{p}^{\mathrm{T}}\mathbf{A}\mathbf{p} + \mathbf{p}^{\mathrm{T}}\mathbf{V} \rightarrow \min$$

$$\begin{aligned} \mathbf{V}' &= \mathbf{A}\mathbf{p} + \mathbf{V} \\ \mathbf{V}' &\geq -e\mathbf{V} \\ \mathbf{p} &\geq \mathbf{0} \end{aligned} \tag{4.17}$$

これにより、線形相補性問題は二次計画問題へと変換され、既存の二次計画法を適用することで撃力 **p** を求めることができる。

本研究では二次計画法として MATLAB の内装関数である quadprog を用いた。

4.3.2 二次計画問題

一般的に、二次計画問題とは次の最小化問題を指す。

$$\begin{aligned} \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x} + \mathbf{f}^{\mathrm{T}}\mathbf{x} &\rightarrow \min \\ \mathbf{A}\mathbf{x} &\leq \mathbf{b} \\ \mathbf{A}_{\mathrm{eq}}\mathbf{x} &= \mathbf{b}_{\mathrm{eq}} \\ \mathbf{lb} \leq \mathbf{x} &\leq \mathbf{ub} \end{aligned} \tag{4.18}$$

以上の式を満たす **x** を求める。ここで **H** は正値対称である必要がある。本研究では制約式は不等号のみであるため、等号の制約式は用いない。

4.4 本研究における物理シミュレーションの流れ

4.4.1 本研究における剛体運動シミュレーションのアルゴリズム概要

ここまで、衝突判定手法としてボクセルベース法を、撃力計算手法として制約ベース法を提案した。これらの手法を組み合わせた本研究の剛体運動シミュレーションのアルゴリズムはFig. 4-9のようになる。

前処理の段階では、物体のメッシュデータからボクセルデータおよび描画のためのポリゴンデータ、慣性テンソルのデータを作成しておく。これらはファイル化してセーブしておき、シミュレーションの初期設定の段階でロードすることでデータを取り込む。

シミュレーションの初期設定の段階では、基本的にシミュレーションによって変化しない値の設定を行う。具体的には時間に関する条件、物体の初期条件および、シミュレーション内の環境条件を設定する。シミュレーションで用いたいボクセル分割レベルもこの時点で設定する。この際に前処理で作成したボクセルデータのレベルより大きい値にしないよう注意が必要である。

以降はタイムステップをパラメータとしたループ計算である。衝突が生じなければ撃力計算を行わずにそのまま重力などを考慮してパラメータを更新し、衝突が生じた場合には制約ベース法を用いて撃力を計算したあとにパラメータを更新する。

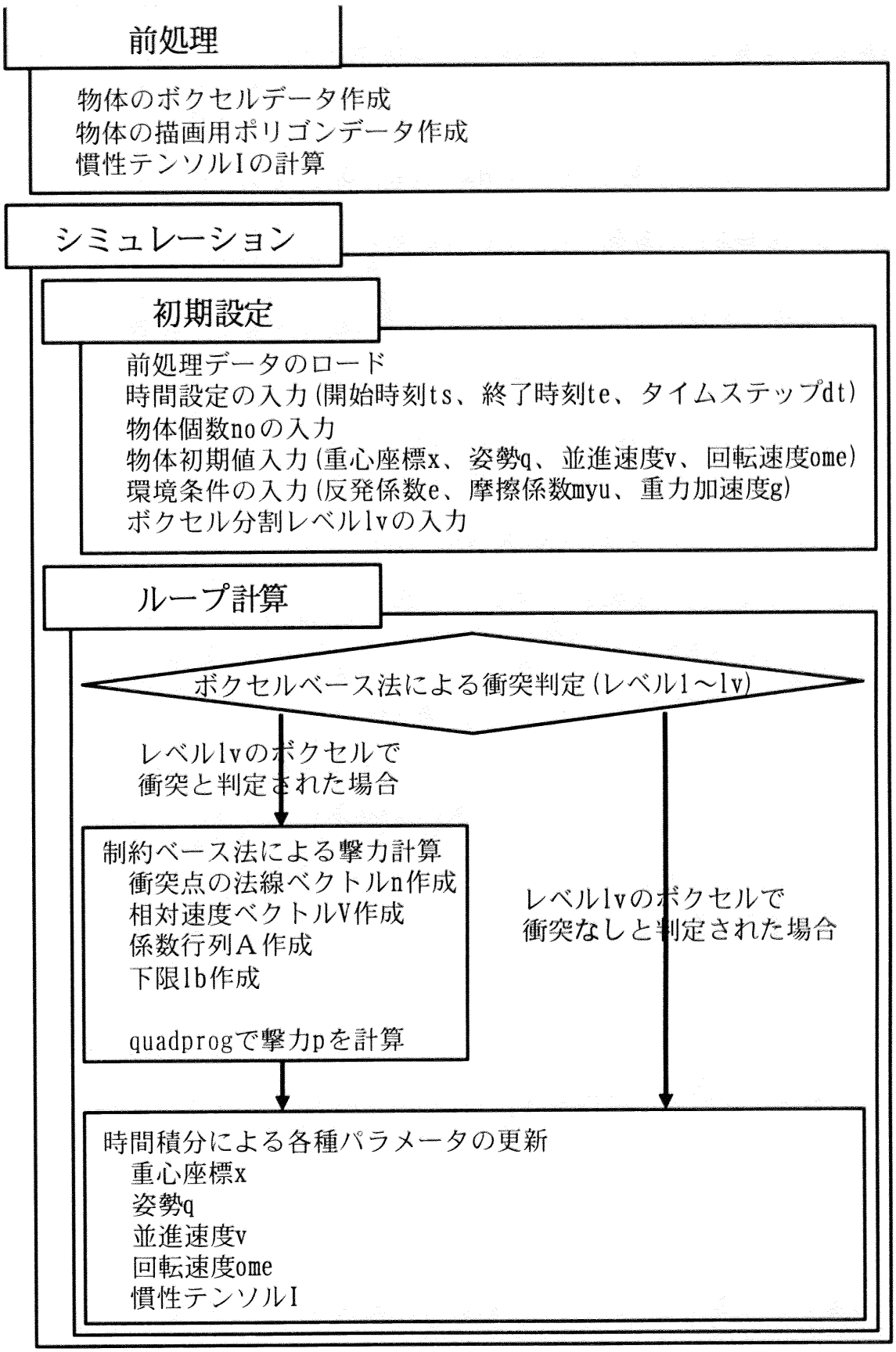

Fig. 4-9 本研究における剛体運動シミュレーションのアルゴリズム

4.4.2 時間積分

ここで扱うような衝突が頻繁におこる問題に対しては衝突による不連続性が発生するため、Runge Kutta 法、Newmark β 法等の高次の積分法を用いるよりも、タイムステップを十分に小さくすることが重要である。そこで、時間積分は陽的オイラー法により行う。時間積分によって更新する物体のパラメータは、重心座標 **x**、姿勢 **q**、並進速度 **v**、回転速度 ω、慣性テンソル **I** である。各パラメータは、以下の式で更新を行う。

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \mathbf{v}_t \Delta t$$

$$\mathbf{q}_{t+\Delta t} = \Delta\mathbf{q} \times \mathbf{q}_t$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \frac{\sum_i \mathbf{p}}{m} \tag{4.19}$$

$$\boldsymbol{\omega}_{t+\Delta t} = \boldsymbol{\omega}_t + \mathbf{I}_t^{-1}\left(\sum_i \mathbf{r}_i \times \mathbf{p}_i\right)$$

$$\mathbf{I}_{t+\Delta t} = \mathbf{R}_t \mathbf{I}_0 \mathbf{R}_t^{\mathrm{T}}$$

ここで、姿勢を管理する **q** はクォータニオンと呼ばれるものである。クォータニオンについては次項に示す。

4.4.3 クォータニオン

剛体運動シミュレーションにおいて、剛体の姿勢(回転)をオイラー角から計算された回転行列で表すと、シミュレーションが進むにつれて数値誤差が増大し、軸周りの回転に伴う誤差が大きくなってしまうなどの不都合が生じる。また、オイラー角には特異点というものが存在し、この点に近づくと時間変化が表現できなくなり、剛体が異常な挙動を示してしまうという問題もある。この問題は、オイラー角を用いる代わりに、ハミルトンによって考案されたクォータニオンによって姿勢を表現することで解決する。

クォータニオンは CG の分野における姿勢の表現法としては一般的に普及しているものであり、本研究でも姿勢の表現にはクォータニオンを用いることとする。

クォータニオンは四元数とも呼ばれ、1 つのスカラーと 3 つのベクトル成分あわせて 4 成分を持ち、次のように表現される。

$$\mathbf{q} = [s, \mathbf{v}] = s + v_x i + v_y j + v_z k \tag{4.20}$$

ここで、i、j、k は異なる虚数単位で、次のような性質がある。

$$\begin{cases} i^2 = -1 \\ j^2 = -1 \\ k^2 = -1 \\ ij = -ji = k \\ jk = -kj = i \\ ki = -ik = j \end{cases} \tag{4.21}$$

クォータニオンは、任意軸周りの回転を表現することが可能である。軸 $\mathbf{v}_{\text{axis}}$ 周りに θ だけ回転させるクォータニオンは以下のように定義される。

$$\mathbf{q}(\theta) = \left[\cos(\theta/2), \mathbf{v}_{\text{axis}} \sin(\theta/2)\right] \tag{4.22}$$

また、クォータニオンが与えられたとき、回転行列は以下のように定義され、クォータニオンと回転行列は相互に変換が可能である。

$$\mathbf{R} = \begin{pmatrix} 1-2v_y^2-2v_z^2 & 2v_xv_y-2sv_z & 2v_xv_z+2sv_y \\ 2v_xv_y+2sv_z & 1-2v_x^2-2v_z^2 & 2v_yv_z-2sv_x \\ 2v_xv_z-2sv_y & 2v_yv_z+2sv_x & 1-2v_x^2-2v_y^2 \end{pmatrix} \tag{4.23}$$

クォータニオンによる物体の回転の表現は、角速度を用いて表現する。角速度 $\boldsymbol{\omega}(t)$ は、物体がその時刻において軸を $\dfrac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}$ として、$|\boldsymbol{\omega}(t)|$ [rad/s]で回転していることを表すとする。 ある時刻 t において角速度が $\boldsymbol{\omega}(t)$ であるとすると、衝突が起こらない場合は次のタイムステップにおいても角速度は変化せず $\boldsymbol{\omega}(t+\Delta t) = \boldsymbol{\omega}(t)$ である。よって、この Δt の間に物体は軸を $\dfrac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}$ として、$|\boldsymbol{\omega}(t)| \cdot \Delta t$ 回転する。この回転を表すクォータニオン $\Delta\mathbf{q}$ は、以下のように表すことができる。

$$\Delta\mathbf{q} = \left[\cos\left(\frac{|\boldsymbol{\omega}(t)| \cdot \Delta t}{2}\right), \frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}\sin\left(\frac{|\boldsymbol{\omega}(t)| \cdot \Delta t}{2}\right)\right] \tag{4.24}$$

したがって、次のステップにおけるクォータニオン $\mathbf{q}(t+\Delta t)$ は以下のように表される。

$$\mathbf{q}(t+\Delta t) = \Delta\mathbf{q} \times \mathbf{q}(t) \tag{4.25}$$

ただし、クォータニオンの積は通常の演算とは異なる。クォータニオンの積は以下のように定義される。

$$[s_1, \mathbf{v}_1] \times [s_2, \mathbf{v}_2] = [s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2] \quad (4.26)$$

式(4.26)の右辺における・はベクトルの内積、×はベクトルの外積を表す。

4.4.4 タイムステップ Δt の決定

時間積分の際、タイムステップは衝突に対する精度および安定性より決定することになる。ポリゴンによる衝突判定においては、このタイムステップの決定が難しく、これを大きく取りすぎると物体同士が貫入してしまうことがあるが、経験的な値を使うしかなく、明確な指針が存在しない。そのため、しばしば不安定な解が生じることがある。一方ボクセルによる手法は、ボクセルのレベル、速度、角速度によってタイムステップを決定することができる。

タイムステップ Δt は、物体のボクセル外接球が貫通しない程度に小さな値をとる必要がある。Fig. 4-10に示すように、物体の速度が速い場合にボクセル外接球同士が貫通してしまう可能性が生じる。このような場合は、Δt の間にボクセル外接球が進む距離が球の半径を超えないことが貫通を生じさせないための条件である。

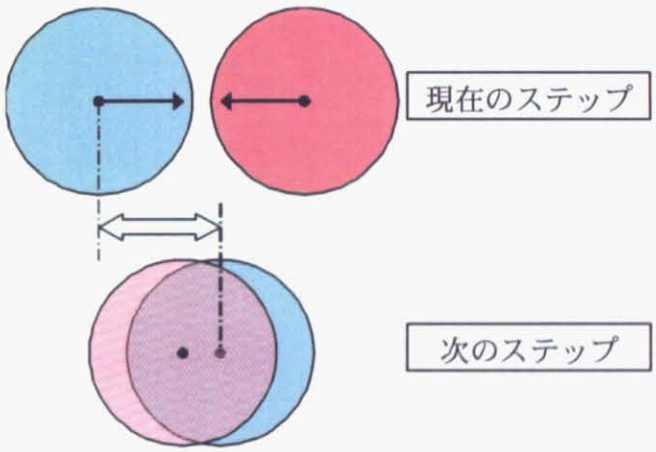

Fig. 4-10 ボクセル外接球同士が貫通してしまう状況

上記の条件は、当該時刻における最大速度 $v_{\max}$ とタイムステップ Δt 、およびボクセル外接球の半径 r を用いて表現することができる。

$$v_{\max} \cdot \Delta t \le r \quad (4.27)$$

最大速度は、物体の重心速度 $\mathbf{v}$ と角速度 ω から計算する。角速度を考えると、物体上で最も大きい速度を持つ点は、重心から最も離れた点だということがわかる。したがって、ボクセルレベル1の外接球の直径を l として最大速度は次のようになる。

$$v_{\max} = |\mathbf{v}| + \frac{l}{2}|\boldsymbol{\omega}| \tag{4.28}$$

また、あるレベルのボクセル外接球の半径 r は l とボクセルレベル lv から求められる。以上のことから、ボクセル外接球が貫通しないためのタイムステップの条件は、以下のようになる。

$$\Delta t \le \frac{l}{2^{lv} \cdot \left(|\mathbf{v}| + \frac{l}{2}|\boldsymbol{\omega}| \right)} \tag{4.29}$$

ただし、実際にシミュレーションを行う場合には、各種の不確定要素を考慮して、式(4.29)より十分に小さな値のタイムステップ Δt を用いることが望ましい。

第5章 見かけの衝突点を用いた簡略化手法

5.1 概要

ボクセルベース衝突判定法を用いた剛体運動シミュレーションのための撃力計算手法として、制約ベース法を採用することを前章で決定したが、その際のシミュレーションによる検証で制約ベース法は衝突点が多くなると、二次計画法による撃力計算の段階で計算時間の多くを費やしていることが判明した。これは制約ベース法が複数の衝突点に対して各撃力の影響を連成させて解くという手法のため、当然生じることではある。そこで、何らかの手法を用いて衝突点を減らすことで撃力計算の負担を軽減できないかと考えた。

本研究では、撃力計算における負担の軽減のために見かけの衝突点を用いた簡略化手法を提案する。見かけの衝突点を求めるためには、物体間、あるいは物体-床間の撃力について仮定をおき、そこからモーメントの釣り合う点に衝突点を集中させ、そこであらためて制約ベース法から計算される撃力を用いて並進速度および回転速度の更新を行うというものである。

見かけの衝突点は、1 つの物体と 1 つの物体との衝突、あるいは 1 つの物体と床との衝突において衝突点をひとつに集約するものである。したがって、ひとつの物体が 3 つの物体と衝突している場合には、見かけの衝突点は各物体間の衝突ごとに計算するため、見かけの衝突点は 3 点となる。

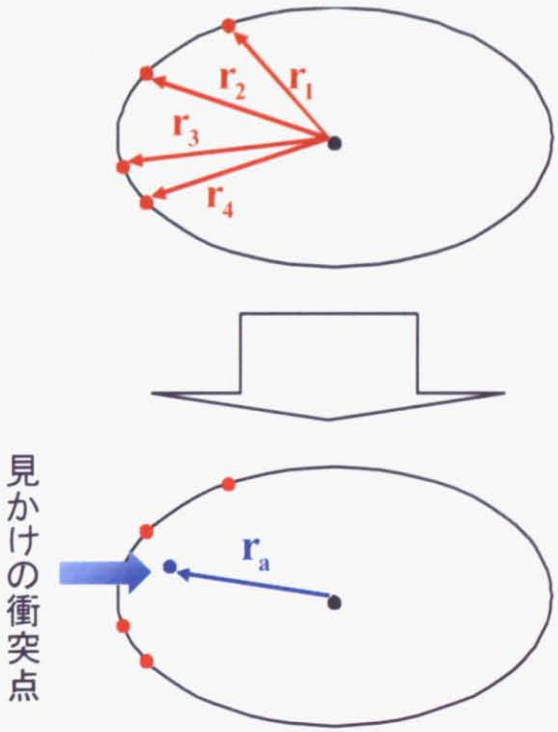

Fig. 5-1 見かけの衝突点のイメージ図

5.2 見かけの衝突点の計算

5.2.1 撃力 p の仮定

見かけの衝突点を求める前に、撃力 **p** の仮定を行わなければならない。力学的にも適切な撃力 **p** を仮定するために、Fig. 5-2のような 2 点衝突の簡易な例で考察を行ってみた。

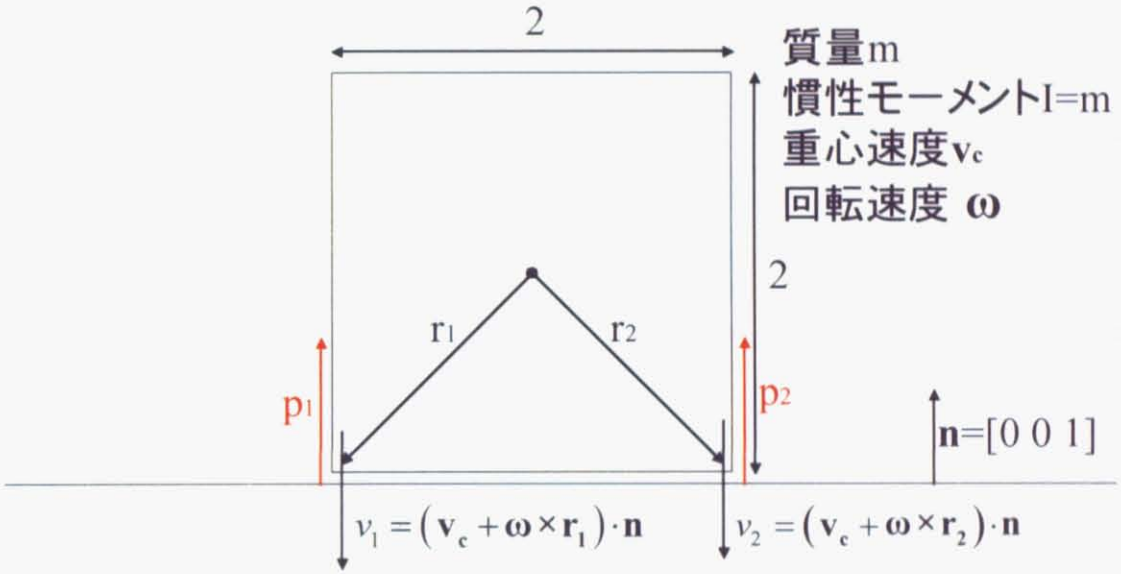

Fig. 5-2 二次元における 2 点衝突の例

このとき、衝突後の相対速度 **V'**は、衝突前の相対速度 **V** と衝突時の撃力 **p** および物体の各種条件を用いて、

$$\mathbf{V}' = \begin{pmatrix} v_1' \\ v_2' \end{pmatrix} = \begin{pmatrix} a+b & -a+b \\ -a+b & a+b \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \mathbf{A}\mathbf{p} + \mathbf{V}$$

$$a = \frac{((\mathbf{r}_1 \times \mathbf{n}) \times \mathbf{r}_1) \cdot \mathbf{n}}{I} = \frac{1}{m} \qquad (5.1)$$

$$b = \frac{1}{m}$$

となる。

ここで、衝突前の相対速度 **V** の値によって、衝突時の撃力 **p** がどのようになるかを計算してみる。簡易化のため、反発係数 e=0.5 とする。

1.　$v_1 = v_2 = -v$ のとき

衝突後の相対速度 **V'**は制約式より、

$$\mathbf{V'} = \begin{pmatrix} v_1' \\ v_2' \end{pmatrix} \geq -e\mathbf{V} = \begin{pmatrix} -0.5v \\ -0.5v \end{pmatrix} \tag{5.2}$$

であればよい。また、制約ベース法の仮定より衝突点に撃力が生じる場合は、その衝突点においては式(5.2)では等号が成立することになる。今回の例の場合、一方の衝突点の撃力はもう一方の衝突点の衝突後の速度に影響を与えない(係数行列 **A** の対角以外の項を見れば明らか)ため、どちらの衝突点においても撃力は生じることとなる。したがって、式(5.2)は等号となり、$v_1' = v_2' = -0.5v$ であることがわかる。

このとき、

$$\mathbf{p} = \mathbf{A}\left(\mathbf{V'} - \mathbf{V}\right) \tag{5.3}$$

より、

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} 0.75mv \\ 0.75mv \end{pmatrix} \tag{5.4}$$

と求められる。

2.　$v_1 = -2v, v_2 = -v$ のとき

この場合も、どちらの点においても衝突時に撃力が発生するので、衝突後の相対速度 **V'**は、

$$\mathbf{V'} = \begin{pmatrix} v_1' \\ v_2' \end{pmatrix} = -e\mathbf{V} = \begin{pmatrix} -v \\ -0.5v \end{pmatrix} \tag{5.5}$$

となる。

1 の場合と同様に、撃力 **p** は、

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} 1.5mv \\ 0.75mv \end{pmatrix} \tag{5.6}$$

以上の 2 つの例を見ると、衝突時の撃力は衝突前の相対速度に比例していることがわかる。この例は 2 点衝突という非常に簡単な例ではあるが、衝突時の撃力が衝突前の相対速度に比例するという考え方は、物理的にも非常に自然なものである。何故ならば、質点を考えると衝突時の撃力は運動量-*mv* を *emv* に変化させる力積であるので、衝突前の相対速度に比例するからである。

そこで本研究では、衝突時の撃力 **p** を、

$$\mathbf{p} \propto \mathbf{V} \tag{5.7}$$

と仮定し、以下この仮定を出発点として見かけの衝突点を用いたシミュレーションの定式化を行っていく。

5.2.2 見かけの衝突点の計算

見かけの衝突点は、仮定した衝突時の撃力 $\mathbf{p}$ に対して、モーメントが釣り合う点を計算することで求める。ある物体が k 個の衝突点で衝突しているとすると、見かけの衝突点 $\mathbf{r}_a$ は、

$$\mathbf{r}_a = \frac{\sum_{i=1}^{k} v_i \mathbf{r}_i}{\sum_{i=1}^{k} v_i} \tag{5.8}$$

で求められる。衝突点 i の相対速度 v_i は撃力 p_i に比例するので、式(5.8)は撃力 $\mathbf{p}$ のモーメントが釣り合う点に一致する。Fig. 5-2の 1 番目の例で見かけの衝突点を計算すると $\mathbf{r}_a = [0, 0, -1]$(Fig. 5-3)、2 番目の例で計算すると、$\mathbf{r}_a = [-1/3, 0, -1]$(Fig. 5-4)となる。今回は、凸形状物体での物体を例としたが、非凸形状の物体の場合は、見かけの衝突点が物体の外側、つまり物体が存在しない点になる可能性もある。

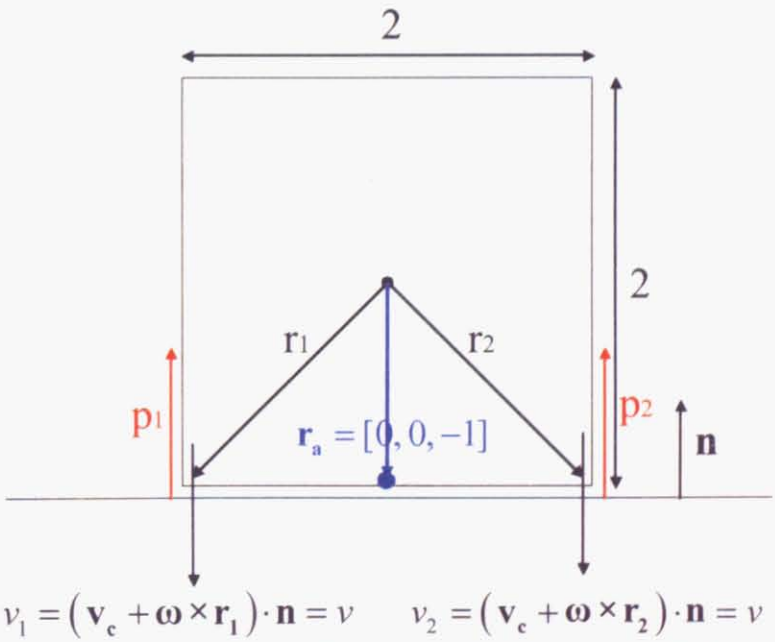

Fig. 5-3 Fig. 5-2の 1 番目の例における見かけの衝突点

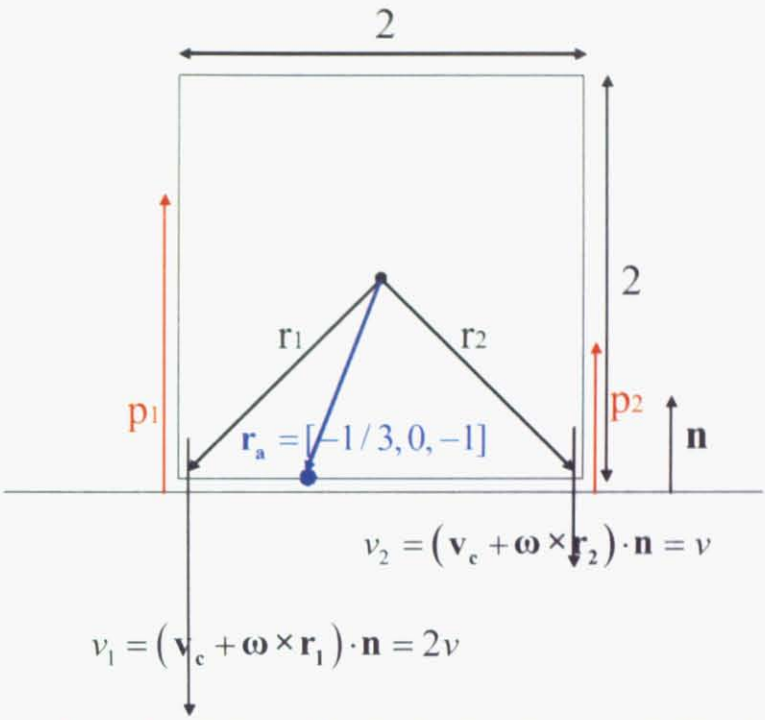

Fig. 5-4 Fig. 5-2の 2 番目の例における見かけの衝突点

5.2.3 見かけの衝突点における法線ベクトルの計算

衝突点の位置を求めると共に、撃力の方向を示す衝突点における法線方向ベクトルを新たに求める必要がある。そこで、撃力 **p** が衝突前の相対速度 **V** に比例するという仮定から出発して見かけの衝突点における法線ベクトルの計算を行う。

$$\mathbf{p} = \alpha \mathbf{V} \tag{5.9}$$

見かけの衝突点での撃力によって使わない場合と一致する物体挙動を実現するには、各衝突点での撃力ベクトルの和と見かけの衝突点における撃力ベクトルが等しくならなければならない。見かけの衝突点における単位法線ベクトルを $\mathbf{n}_a$、撃力の大きさを p_a とすると、

$$\begin{aligned} p_a \mathbf{n}_a &= \sum_i p_i \mathbf{n}_i \\ &= \alpha \sum_i v_i \mathbf{n}_i \end{aligned} \tag{5.10}$$

と表すことができる。

ここで、$\mathbf{n}_a$ は単位ベクトルなので、

$$\mathbf{n}_a = \frac{\sum_i v_i \mathbf{n}_i}{\left|\sum_i v_i \mathbf{n}_i\right|} \tag{5.11}$$

である。同様に p_a は、

$$p_a = \alpha \left|\sum_i v_i \mathbf{n}_i\right| \tag{5.12}$$

となる。

物体と床との衝突においては、見かけの衝突点は式(5.8)を用いて物体と物体の衝突と同様に計算を行うが、法線方向ベクトルについては式(5.11)からも明らかなように、どの衝突点 i においても $\mathbf{n}_i = [0,0,1]$ である。したがって $\mathbf{n}_a = [0,0,1]$ となるため、計算を省略することができる。

5.2.4 見かけの衝突点での相対速度

撃力計算の際には、衝突点座標および法線方向ベクトルのほかに相対速度の情報が必要となる。見かけの衝突点における相対速度は以下のようにして求められる。

物体 A と物体 B が複数点で衝突したとすると、見かけの衝突点はそれぞれ $\mathbf{r}_{Aa}$、$\mathbf{r}_{Ba}$ と表される。$\mathbf{r}_{Aa}$ における物体 A の速度は、

$$\mathbf{v}_{Aa} = \mathbf{v}_A + (\boldsymbol{\omega}_A \times \mathbf{r}_{Aa}) \tag{5.13}$$

同様に、$\mathbf{r}_{Ba}$ における物体 B の速度は、

$$\mathbf{v}_{Ba} = \mathbf{v}_B + (\boldsymbol{\omega}_B \times \mathbf{r}_{Ba}) \tag{5.14}$$

となる。したがって物体 A の見かけの衝突点での相対速度は、

$$v_{Aa} = (\mathbf{v}_{Aa} - \mathbf{v}_{Ba}) \cdot \mathbf{n}_{Aa} \tag{5.15}$$

同様に、物体 B の見かけの衝突点での相対速度は、

$$v_{Ba} = (\mathbf{v}_{Ba} - \mathbf{v}_{Aa}) \cdot \mathbf{n}_{Ba} \tag{5.16}$$

と表すことができる。

5.3 見かけの衝突点を用いたシミュレーション

5.3.1 見かけの衝突点を用いた場合の時間積分

見かけの衝突点と法線方向ベクトル、および相対速度を用いて求めた撃力を従来の時間積分の式に適用して次のステップにおけるパラメータを決めようとすると問題が生じる場合がある。

問題は、見かけの衝突点における撃力の法線方向ベクトル周りに回転速度があるという状況で生じる(Fig. 5-5、物体が床の法線ベクトル方向周りに回転して落下していく例)。

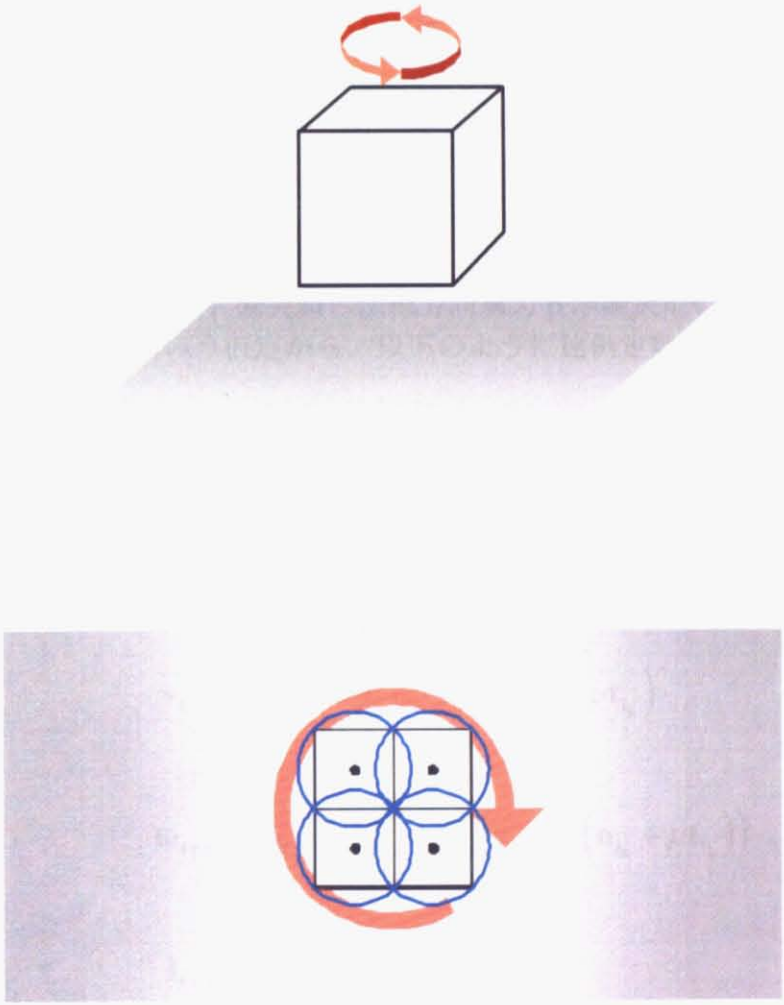

Fig. 5-5 見かけの衝突点を用いると問題が生じる衝突の例(レベル 3)
上：横から見た図
下：下から見た図

このとき、見かけの衝突点は重心直下の点となり、法線方向ベクトルは$\mathbf{n}_a=[0,0,1]$となる。こうして求められた撃力 $\mathbf{p}$ を時間積分の式に代入すると、摩擦が生じない場合は全く問題なく更新できるが、摩擦が生じる場合においては衝突点が重心直下に集約されているため、回転に対する摩擦力が働かず、回転速度が減衰していかないという現象が起こってしまう。

この問題を解決するために、見かけの衝突点を用いた場合の時間積分についてあらためて定式化する必要がある。見かけの衝突点を用いない場合の時間積分の式は第4章でも示したように、以下のようになる。

$$\begin{aligned}
\mathbf{x}_{t+\Delta t} &= \mathbf{x}_t + \mathbf{v}_t \Delta t \\
\mathbf{q}_{t+\Delta t} &= \Delta\mathbf{q} \times \mathbf{q}_t \\
\mathbf{v}_{t+\Delta t} &= \mathbf{v}_t + \frac{\sum_i \mathbf{p}}{m} \\
\boldsymbol{\omega}_{t+\Delta t} &= \boldsymbol{\omega}_t + \mathbf{I}_t^{-1}\left(\sum_i \mathbf{r}_i \times \mathbf{p}_i\right) \\
\mathbf{I}_{t+\Delta t} &= \mathbf{R}_t \mathbf{I}_0 \mathbf{R}_t^{\mathrm{T}}
\end{aligned} \tag{5.17}$$

式(5.17)を今までと同様に、衝突時の法線方向撃力 $\mathbf{p}$ が衝突前の法線方向相対速度 $\mathbf{V}$ に比例する($\mathbf{p} \propto \mathbf{V}$)という仮定から、以下のように比例定数$\alpha_k$を用いて書き換えることができる。

$$\begin{aligned}
\mathbf{x}_{t+\Delta t} &= \mathbf{x}_t + \mathbf{v}_t \Delta t \\
\mathbf{q}_{t+\Delta t} &= \Delta\mathbf{q} \times \mathbf{q}_t \\
\mathbf{v}_{t+\Delta t} &= \mathbf{v}_t + \frac{1}{m}\sum_k \alpha_k \sum_{i_k} v_{i_k}\left(\mathbf{n}_{\mathbf{i}_k} + \mu \mathbf{t}_{\mathbf{i}_k}\right) \\
\boldsymbol{\omega}_{t+\Delta t} &= \boldsymbol{\omega}_t + \mathbf{I}_t^{-1}\sum_k \alpha_k \sum_{i_k}\left(v_{i_k}\mathbf{r}_{\mathbf{i}_k} \times \left(\mathbf{n}_{\mathbf{i}_k} + \mu \mathbf{t}_{\mathbf{i}_k}\right)\right) \\
\mathbf{I}_{t+\Delta t} &= \mathbf{R}_t \mathbf{I}_0 \mathbf{R}_t^{\mathrm{T}}
\end{aligned} \tag{5.18}$$

ここでkは対象物体に衝突している物体のインデックス、i_kは物体kとの衝突点のインデックスを表す。また、μは摩擦係数、$\mathbf{t}_{\mathbf{i}_k}$は物体$k$との衝突点$i_k$における撃力の接線方向単位ベクトルを表す。

比例定数α_kは、式(5.12)より、

$$\alpha_k = \frac{p_{a_k}}{\left| \sum_{i_k} v_{i_k} \mathbf{n}_{i_k} \right|} \tag{5.19}$$

と表すことができる。この際の p_{a_k} は、見かけの衝突点、見かけの衝突点における法線方向ベクトル、および相対速度から制約ベース法を用いて求めた値である。

撃力 **p** を仮定したあとにあらためて撃力を求める理由は、複数物体との衝突が生じた際に撃力が連成されるためである。また、衝突している物体ごとに衝突点を集約させると、グローバル座標における各物体間の見かけの衝突点座標は一致する。

5.3.2 見かけの衝突点を用いたシミュレーションの流れ

以上、見かけの衝突点を用いた簡略化手法におけるシミュレーションの定式化を行ってきた。シミュレーション全体の流れはFig. 5-6のようになる。

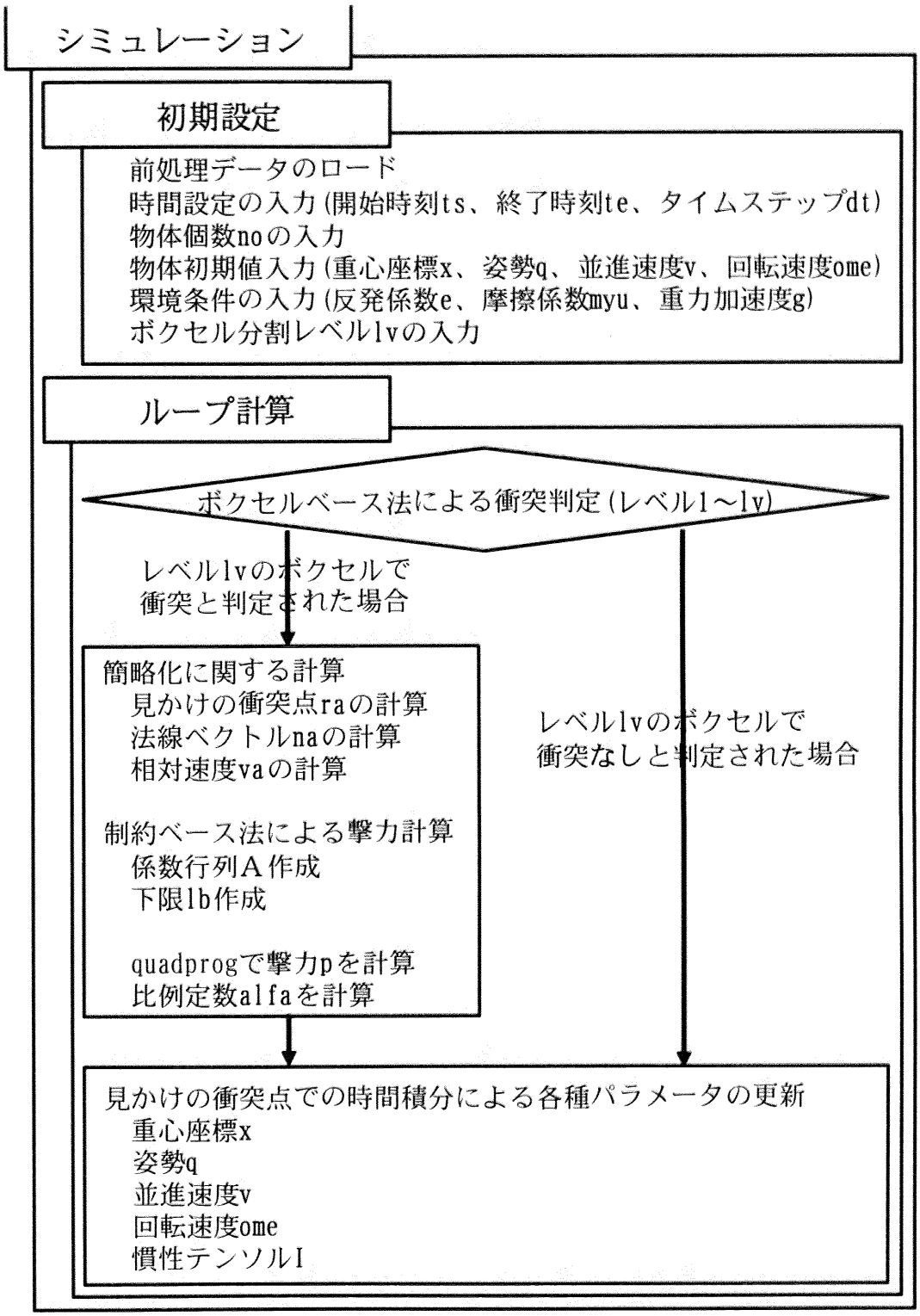

Fig. 5-6 見かけの衝突点を用いた簡略化手法におけるシミュレーションの流れ