2016 年度　卒　業　論　文

有限要素法を用いた
紐帯のリアルタイム力学シミュレーションに関する研究

指導教員：渡辺 大地 講師

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0113193
後藤 佳樹

2017 年 3 月

2016 年度　　卒　業　論　文　概　要

論文題目

有限要素法を用いた
紐帯のリアルタイム力学シミュレーションに関する研究

メディア学部 学籍番号：**M0113193**	氏 名	後藤 佳樹	指導 教員	渡辺 大地 講師

キーワード	弾性体、リアルタイム、シミュレーション、有限要素法、紐帯

近年、コンピュータの性能の向上によりゲーム等のエンタテイメントコンテンツにおいて物理処理をリアルタイムに扱えるようになった。しかし、弾性体シミュレーションはリアルタイム処理が難しいとされ、インタラクティブコンテンツへの適用例でも簡易モデルを利用したアニメーションのみの実現に焦点が当てられたものが多かった。本論文ではインタラクティブコンテンツ向けに紐帯のリアルタイムな物理シミュレーション機構を作成した。

既存のインタラクティブコンテンツ向けの弾性体シミュレーションにはバネ・質点系を用いたモデルが採用されることが多い。しかし、バネ・質点系では外力がなどにより大きな変形が発生した際に質点が発散するような挙動をしたり、変形が意図しないほどに大きくなってしまうことがある。

そこで本研究では 2 次元弾性体に対象を絞り、弾性体のみの変形処理に加えて、自己衝突や他の物体との衝突を検出してリアルタイムに計算する手法を提案する。本手法では三角形メッシュを 1 方向に連続配置したものを紐帯のモデルとして扱う。紐帯モデルの質点の操作による変位や、弾性体の自己衝突とプリミティブな形状の剛体との衝突によって生じた変位を、有限要素法とオイラー陽解法を併用することでリアルタイム計算による変形処理を実現した。そして、本手法を用いたアプリケーションを作成し、ユーザーの入力や衝突による変形をリアルタイムに計算できるかを検証した。その結果、本手法を用いることで外力による変形や衝突による変形をリアルタイム計算で実現できることを示した。

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 研究背景と目的

近年、コンピュータの性能の向上により家庭用ゲーム機やノートパソコンなどのコンピュータでも物理演算をリアルタイムに計算できるようになり、ゲーム等のエンタテイメントコンテンツにおいても物理演算を用いられる場合が多くなった。

しかし、物理演算のなかでも弾性体のシミュレーションはリアルタイムに計算することが難しいとされており、インタラクティブコンテンツへの適用例でもアニメーションのみの実現に焦点が当てられたものが多い。それらは、キャラクターの各姿勢における法線マップを予め出力してリアルタイム処理中のボーンの動作に合わせて適用する [1] 等の限定的な用途であることが多い。一方で、物体の衝突などの現象を表現する場合は計算が高速に行えるために剛体を用いる場合が多く、弾性体の衝突現象をリアルタイム計算で表現した事例は少ない。

弾性体のシミュレーションに関する手法や研究は多数存在する。オープンソースの物理エンジンである Bullet Physics Library[2] では弾性体の表現にバネ・質点系を用いる。バネ・質点系モデルは処理速度こそ速いが、紐帯のようにいくつもの質点が連なって接続された場合、質点に大きな力を与えて局所的に大きな変形が発生した際に形状が破綻を起こす。破綻を起こしたモデル

では一部の質点が不自然な振動をしてしまう。これはバネ・質点系モデルでは隣接点との位置関係しか考慮しないため [3] である。

小山ら [4] は Shape Matching 法を拡張した手法により、基本形状とアーティストが望む例示の形状を与えることで直感的かつリアルタイムな弾性体の変形を実現した。しかし、ゲーム等のコンテンツにおいては、プレイヤーはオブジェクトを押す・引く等モデルを直接操作することが多い。

永安ら [5] はバネ・質点系モデルの縫合糸モデルを運動方程式適用後に拘束点を考慮して補正することで、縫合糸の不自然な伸びの無い縫合糸のモデルを実現した。しかし運動方程式を解いた後に補正を行うために、接触対象と縫合糸とが正しく相互作用しない問題点がある。

中島ら [6] は有限要素法とバネ・質点系モデルを併用して安定した弾性体のリアルタイム計算を実現し、有限要素法の計算過程で算出された反力を力覚デバイスに反映させた。この手法では弾性体の変形をリアルタイム計算を実現したものの、自己衝突や他の物体との衝突は考慮していない。

本研究では 2 次元弾性体に対象を絞り、弾性体のみの変形処理に加えて、自己衝突や他の物体との衝突を検出してリアルタイムに計算する手法を提案する。紐帯の形状モデルとして三角形メッシュを 1 方向に連続配置してモデル化した。質点の操作による変位や、弾性体の自己衝突とプリミティブな形状の剛体との衝突によって生じた質点の変位を、有限要素法とオイラー陽解法を併用することでリアルタイムに計算して変形処理を実現できた。提案手法の妥当性を検証するために、本手法を用いたアプリケーションを実装してその有用性を示した。

1.2 本論文の構成

本論文の構成は以下のとおりである。第 2 章では本研究で取り扱う弾性体と紐帯の性質と、そのシミュレーション手法について述べる。第 3 章では本研究で使用する紐帯のモデルの構造につ

いて説明し、シミュレーションの手法について述べる。第 4 章では本研究で提案したシミュレーション手法を用いた紐帯アプリケーションを用いて検証と考察を行う。第 5 章では本研究の成果と意義をまとめ、今後の展望を述べる。

第 2 章

紐帯について

本章では紐帯の性質と物理シミュレーションにおける紐帯のモデル化について説明する。2.1 節では紐帯が属する弾性体について説明する。2.2 節では弾性体の中でも紐帯に固有な性質とそのシミュレーション手法について説明する。2.3 節では本研究の位置づけと研究方針について述べている。

2.1　弾性体について

物理シミュレーションにおいて、形状が一定である剛体に対して、ゴムや紐帯のように形状が変化する物体のことを弾性体と呼ぶ。

弾性体は変形が一定の範囲において外力が加わることにより形状が変化し、かかっている外力がなくなれば元の状態に戻る弾性変形を行う性質をもっている。変形が一定の範囲を超えると元の形状に戻らなくなる塑性変形を示したり、断裂などの形状の破壊が発生する場合がある。

弾性体のシミュレーション手法の種類は、物理現象に基づく物理ベースのシミュレーションと、アニメーション作成を目的とした幾何学ベースのシミュレーションがある。

物理ベースのシミュレーション法は粒子法と格子法に分類される。格子法は各格子点において

繋がりのある他の格子点との間に関係式を立て、それを解くことで物理的なシミュレーションを実現する。本研究で用いる有限要素法 [7] やバネ・質点系は格子法に分類される。

一方で粒子法は物体を粒子の集合とみなし、各粒子に物理法則を適応することで物理的なシミュレーションを実現する。粒子法には、液体の非圧縮性流体のシミュレーションに適した MPS 法 [8] や、Lucy[9] によって宇宙物理学のために開発された圧縮性流体のシミュレーションに適した SPH 法などがある。

アニメーションを作成することを目的とした手法にはシェイプマッチング法がある。Müller ら [10] はシェイプマッチング法を用いて、メッシュレスな形状変形シミュレーションを実現している。これは物体の形状を構成する頂点に質点を配置し、質点に対して重力等の外力や他の物体との干渉だけを考慮して変形処理を行い、その後バネ・質点系のように引力を適用して、物体の初期形状から回転と平行移動だけで変換した形状にできるだけ近づけるように変形する手法である。海上ら [11] はシェイプマッチング法を用い、形状の破壊を伴う弾性体の変形シミュレーションをリアルタイム計算で実現している。

2.2 紐帯の構造

紐帯とは、繊維と呼ばれる細長い物体を束ねた物体のことを表す。紐帯は繊維自体の物質の性質と束る際の繊維の撚り方、紐帯の太さによって強度や伸長の仕方などの性質が異なり、様々な用途に応じた規格 [12][13] が存在する。また、ワイヤーロープのように複数の紐帯構造を撚り合わせる場合もある。ワイヤーロープは素線と呼ばれる繊維を撚り合わせたストランドを、さらに繊維心に撚り合わせる構造をとる。素線の撚り方やストランドの撚り方を変えることで異なる特徴を持つため、用途に応じて使い分けられている。ワイヤーロープの構造の例を図 2.1 に示す。

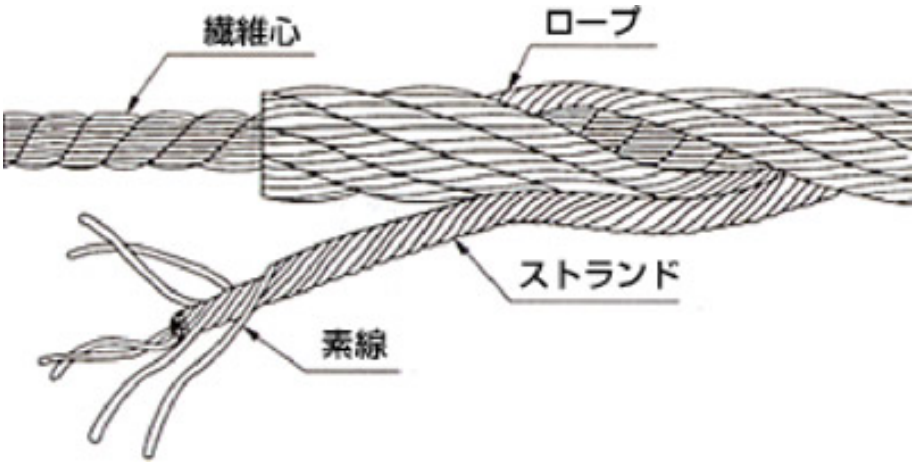

図 2.1　ワイヤーロープの構造

出典：https://www.tokyorope.co.jp/product/wirerope/outline.html

繊維構造を考慮した弾性体のシミュレーションとして、Yarn-Level Simulation[14] がある。Yarn-Level Simulation シミュレーションは繊維ごとの絡まり合いによる繊維の衝突などを考慮して弾性体の伸長を表現できるシミュレーション手法である。しかし、その計算量からリアルアイムシミュレーションには向かないという欠点がある。

2.3　本研究の位置づけと研究方針

前節までに弾性体のシミュレーション手法および、紐帯の構造やその性質について述べた。本研究では、ゲーム等のインタラクティブコンテンツでの利用を想定し、物理現象に基づいて弾性体の変形をリアルタイムにシミュレートできることを目的とした。そのために以下の条件を満たすことを目標とした。

- 物理現象に基づくシミュレーション手法である
- リアルタイム処理が可能である
- 他のオブジェクトと力学的に衝突する
- 安定性あり、外力などにより形状が破綻しにくい

これらの目標を満たす手法を提案し、インタラクティブコンテンツ制作における表現の幅を広げることを目指す。紐帯の変形に関しては、2.2 節で述べた紐帯の複雑な複合構造を考慮して計算をすることは処理速度を考えると実用的ではない。シェイプマッチング法は 2.1 節で述べたように物理ベースのシミュレーション手法ではないので適切ではない。また、粒子法は質点間に接続を持たせないため紐帯モデルのレンダリングの際にメッシュを構成しなければならず、レンダリングコストがかかる。

本研究では紐帯の形状モデルとして三角形メッシュを 1 方向に連続配置したモデルを用い、中島ら [6] の手法を参考にして格子法の手法の一つである有限要素法とバネ・質点系を併用したシミュレーション手法を用いる。紐帯の繊維の撚り方が一定であると仮定すると、有限要素法では剛性行列、バネ・質点系ではバネ係数によって剛性を表現することができるため、これによって紐帯の伸長度を調整することが可能である。さらに質点間の接続情報が保持されるのでメッシュの再構築の必要無しにレンダリングできる。

また、変形範囲の大きさにかかわらず常に弾性変形が発生するものとし、形状の破壊は考えないものとする。これはインタラクティブコンテンツ制作において、塑性や形状の断裂は意図した場合を除き、弾性が消えてしまうために弾性変形を望んで利用する場合の妨げになることを考慮した。

第 3 章

提案手法

本章では本研究で提案する紐帯を表現した弾性体の変形シミュレーションと弾性体の衝突を表現する手法について述べる。3.1 節では紐帯のモデル化手法について述べる。3.2 節では 3.4 節、3.5 節で使用する有限要素法とバネ・質点系の概要について述べる。3.3 節では提案手法における全体の計算処理の流れを示す。以降の節では各処理過程における手法の詳細について述べる。3.4 節、3.5 節では紐帯モデルの変形の計算手法について述べる。3.6 節では紐帯モデルにおける自身の一部との衝突を検出し変位させる手法について述べる。3.7 節では紐帯モデルと他の物体との衝突を検出し、両者のオブジェクトの衝突による変位を計算する手法について述べる。

3.1　紐帯のモデル化

3.1.1　紐帯モデルのデータ構成

本研究では特定の質点を接続して構成した三角形を弾性体の最小単位とし、弾性体の要素と呼ぶ。弾性体の要素の情報は、後述する弾性体の衝突判定や弾性体モデルのメッシュ描画の際に用いる。図 3.1 にその弾性体の要素を図示する。

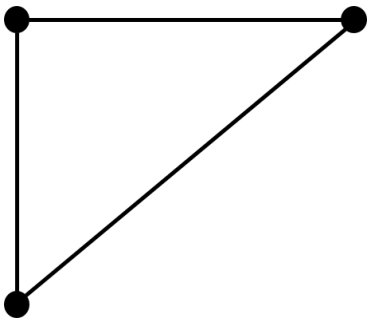

図 3.1　弾性体の要素

また、質点間に接続関係を持たせる。この接続関係は有限要素法の手法中における剛性や、バネ・質点系の補正処理の際に用いるバネの接続を表現する際に用いる

横方向の質点数を $w(w > 1)$、縦方向の質点数を $h(h > 1)$、紐帯モデル中の質点を $\mathbf{P}(i, j)$ $(0 < i \leqq w, 0 < j \leqq h)$ とすると、初期状態で一定間隔に配置された質点群について、質点 $\mathbf{P}(i, j)$ は自身の上下左右と斜方向に隣接した質点、すなわち $\mathbf{P}(i-1, j-1)$、$\mathbf{P}(i, j-1)$、$\mathbf{P}(i+1, j-1)$、$\mathbf{P}(i-1, j)$、$\mathbf{P}(i+1, j)$、$\mathbf{P}(i-1, j+1)$、$\mathbf{P}(i, j+1)$、$\mathbf{P}(i+1, j+1)$(ただし各質点は $0 < i \leqq w$, $0 < j \leqq h$ を満たす) と接続するものとする。横方向の質点数を 3、縦方向の質点数を 3 としたときの紐帯モデルを図 3.2 に表す。

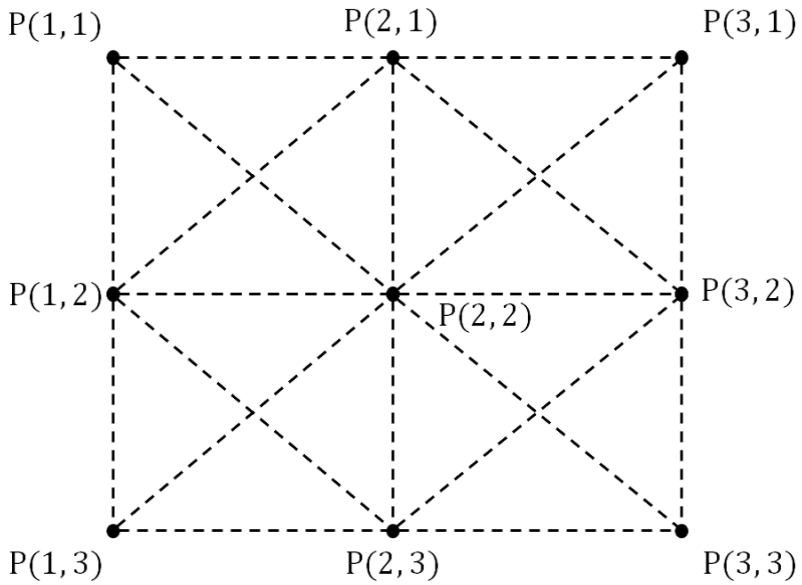

図 3.2　横方向の質点数を 3、縦方向の質点数を 3 としたときの紐帯モデル

3.1.2　境界ボリュームの設定

紐帯モデルが自己衝突や他のオブジェクトと衝突を行えるようにするために、モデルに対して衝突を検出するための領域を設定する必要がある。これを境界ボリューム [15] という。境界ボリュームには円や矩形や三角形などの単純な形状を用い、設定する対象の領域をすべて含むようにする。また、あるシミュレーションステップで衝突を検出した場合、シミュレーションステップが終了した時点でその物体どうしが交差していないことを保証するための処理を衝突応答 [16] という。

弾性体の要素に円形境界ボリュームを設定した場合、円形境界ボリュームどうしの衝突判定と衝突応答を計算することで自己衝突判定を実現できる。円形境界ボリュームは他の形状の境界ボリュームを用いた場合に比べて衝突判定と衝突応答を高速に計算できる。弾性体の要素に円形境界ボリュームを設定例を図 3.3 に示す。

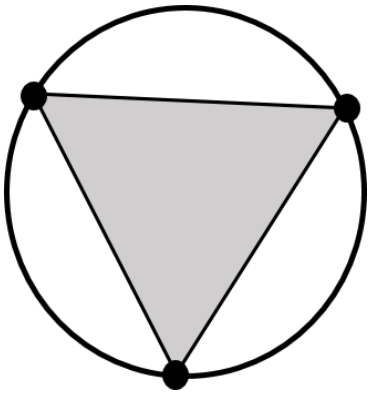

図 3.3　弾性体の要素に対する円形境界ボリュームの設定例

しかし、三角形状の弾性体の要素に対して円形の境界ボリュームを設定すると、境界ボリューム中に弾性体の要素の領域に含まれない部分が発生してしまう。また、形状変形処理後に弾性体の要素の領域をすべて含むように円形境界ボリュームの半径を変えつつ再設定しなければならず、弾性体の要素の形状によっては境界ボリューム中に弾性体の要素の領域に含まれない部分が大き

くなってしまう場合がある。これにより衝突を検出する領域が紐帯モデルよりも大きくなってしまい、その大きさも弾性体の要素の形状によって異なる不安定なものとなる。弾性体の要素の領域に含まれない部分が大きい円形境界ボリュームの例を図 3.4 に示す。

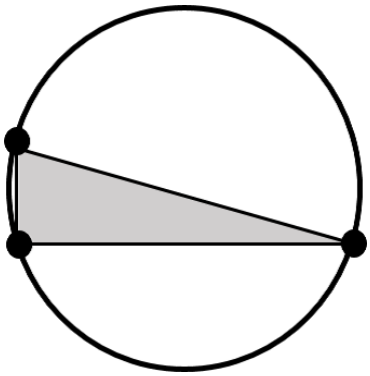

図 3.4　境界ボリューム中の弾性体の要素の領域に含まれない部分が大きくなる場合

一方で弾性体の要素は 3 頂点で構成されるので、弾性体の要素の頂点を三角形境界ボリュームの各頂点とみなすことができる。これによって弾性体の要素と同一の領域の境界ボリュームを構成できる。しかし三角形境界ボリュームどうしの衝突応答処理では、交差を解消するベクトルを求める計算が非常に高価でありリアルタイム性に欠ける。

本手法では、弾性体の要素の頂点である質点に円形の境界ボリューム、弾性体の要素に三角形の境界ボリュームを設定する。これにより円形境界ボリュームを弾性体の要素の形状に合わせて再設定する必要はなく、さらに三角形境界ボリュームどうしの衝突判定、衝突応答よりも高速に計算できる。

図 3.5 はシミュレーション中のある時刻でのスナップショットにおける質点の境界ボリュームと弾性体の要素の境界ボリュームを表している。

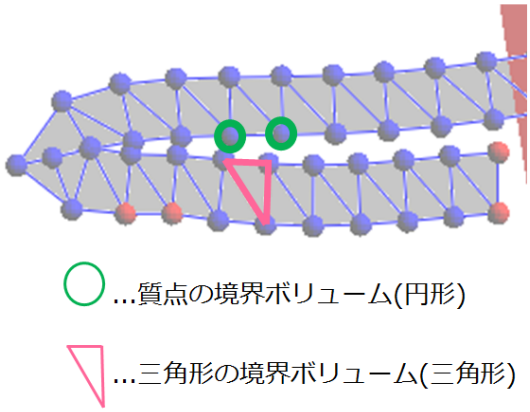

図 3.5　質点と弾性体要素の境界ボリューム

また、円形の境界ボリュームを適用するモデル毎に m_c、三角形の境界ボリュームを適用するモデル毎に $m_t(m_c{>}0,\ m_t{>}0)$ を設定する。この値は後述する 3.6.2 項で交差を解消する際に、弾性体の要素の変位と質点の変位がどの程度優先されるかを算出する際に用いる。

3.2　形状変形の概要

提案手法における形状変形の計算には、有限要素法とバネ・質点系を併用したシミュレーション手法を用いる。

バネ・質点系とは、ある方向における変位によって生じる復元力は、その変位の量に比例するというフックの法則 [17] に基づいて、接続した質点間の距離応じた復元力を発生させることで物体の伸縮を表現する手法である。ただし、接続した質点の距離のみで計算を行うので、局所的な変形が起こることで変形の伝播に遅延が発生し、質点の運動が収束せず発散してしまい著しく形状が破たんすることがある。

有限要素法では、物体を有限の範囲にモデル化して小さな要素に区切り、各要素毎に成り立つ支配方程式を作成する。そして、各要素における支配方程式を、計算対象の全領域分足し合わせることで巨大な連立 1 次方程式を作成し、それを解くことで近似解を求める手法である。有限要

素法ではバネ・質点系とは異なり、質点間での計算ではなく全質点の状態を考慮して変形を計算することができる。一方で、有限要素法は計算処理のなかで用いる行列が非正則なものになると計算を継続することができない。その場合はバネ・質点系による変形のみが行われることによって形状変形の計算を持続することが可能である。

3.3　シミュレーションステップ

本提案手法の計算手順を図 3.6 に示す。

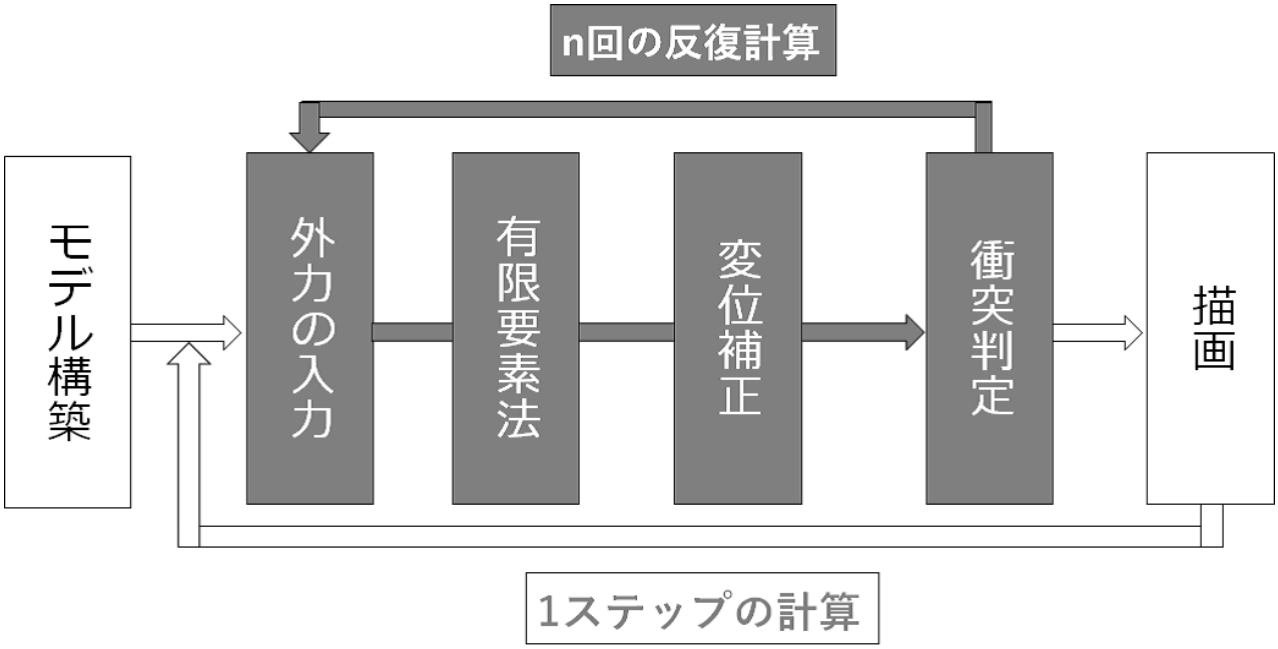

図 3.6　本手法におけるシミュレーションステップ

まず、シミュレーション開始時に 3.1 節の手法に基づいて紐帯のモデル構築を行う。以降、外力の入力から描画までの処理を毎フレーム行う。外力の入力から衝突判定までは 1 フレーム中に任意の回数繰り返すことができ、複数回の反復計算を行うことで精度の良いシミュレーション結果が得られるが、計算量が増えるためリアルタイム計算を維持することが難しくなる。

反復計算の結果、変形移動した頂点に基づいて弾性体モデルのポリゴン表示を更新する。

3.4　有限要素法の適用

ユーザー操作による外力の入力など、直前の計算ステップで生じた質点の変位をふまえて有限要素法を適用する。

3.4.1　要素毎の剛性行列の作成

本モデル中の弾性体の 1 要素について要素の量端点をそれぞれ i、j とするとその要素にかかる力と変位の関係は (3.1) 式で表すことができる。ここで f_i、f_j は両端点の質点にかかる力、u_i、u_j は両端点の変位、k は要素 i、j における物性値、右辺の行列は剛性行列を表す。

$$\begin{pmatrix} f_i \\ f_j \end{pmatrix} = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{pmatrix} u_i \\ u_j \end{pmatrix} \tag{3.1}$$

2 次元空間で要素における力や変位を考えるときは、各軸方向の成分ごとに分けて計算する必要がある。i、j におけるローカル座標をそれぞれ $(\overline{u}_{ix}, \overline{u}_{iy})$、$(\overline{u}_{jx}, \overline{u}_{jy})$、それぞれに働く力を $(\overline{f}_{ix}, \overline{f}_{iy})$、$(\overline{f}_{jx}, \overline{f}_{jy})$ とすると (3.2) 式のように表せる。

$$\begin{pmatrix} \overline{f}_{ix} \\ \overline{f}_{iy} \\ \overline{f}_{jx} \\ \overline{f}_{jy} \end{pmatrix} = \begin{bmatrix} k & 0 & -k & 0 \\ 0 & 0 & 0 & 0 \\ -k & 0 & k & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \overline{u}_{ix} \\ \overline{u}_{iy} \\ \overline{u}_{jx} \\ \overline{u}_{jy} \end{pmatrix} \tag{3.2}$$

さらに (3.2) 式をまとめて

$$\overline{\mathbf{F}} = \mathbf{K}\overline{\mathbf{U}} \tag{3.3}$$

とする。

グローバル座標での力成分を (f_{ix}, f_{iy})、(f_{jx}, f_{jy})、グローバル座標からローカル座標への座標変

換行列 $\mathbf{T}$ とするとローカル座標とグローバル座標での力の成分の関係は (3.4) 式で表せる。

$$\overline{\mathbf{F}} = \mathbf{TF} \tag{3.4}$$

各質点の変位についても同様の関係が成立するため

$$\overline{\mathbf{U}} = \mathbf{TU} \tag{3.5}$$

である。また、(3.4) 式と (3.5) 式に (3.3) 式をあてはめると

$$\mathbf{TF} = \mathbf{KTU} \tag{3.6}$$

となり、この式を整理すると

$$\mathbf{F} = \mathbf{T}^{-1}\mathbf{KTU} \tag{3.7}$$

となる。

以上のことから、紐帯モデルの要素におけるグローバル座標での要素ごとの剛性行列を (3.9) 式のように求めることができる。ここで紐帯モデルの要素におけるローカル座標の x 軸を $\overline{x}$ 軸とし、x 軸と $\overline{x}$ 軸のなす角を $\theta_{x\overline{x}}$ とすると、c、s はそれぞれグローバル座標系の各軸の方向余弦により

$$c = \cos\theta_{x\overline{x}}, s = \sin\theta_{y\overline{x}} \tag{3.8}$$

と表せるので、要素の質点 i、j における剛性行列は (3.9) 式で表せる。

$$\mathbf{T}^{-1}\mathbf{KT} = k \begin{bmatrix} cc & cs & -cc & -cs \\ cs & ss & -cs & -ss \\ -cc & -cs & cc & cs \\ -cs & -ss & cs & ss \end{bmatrix} \tag{3.9}$$

3.4.2　全体剛性行列の作成

(3.9) 式をすべての要素に適用して要素剛性行列を求めそれを足し合わせることで全体剛性行列を作成する。ここで、3 つの質点 A、B、C が B を共有して接続している状況を考える。(3.1) 式と同様にして質点 A、B、C にかかる力をそれぞれ f_a、f_b、f_c、質点の変位をそれぞれ u_a、u_b、u_c とし、接続 AB に関して質点 B にかかる力を f_{b1}、接続 BC に関して質点 BC にかかる力を f_{b2} とすると、AB 間の関係は (3.10) 式、BC 間の関係は (3.11) 式で表せる。(3.10) 式、(3.11) 式における右辺の行列は前項の座標変換を行った剛性行列である。

$$\begin{pmatrix} f_a \\ f_{b1} \end{pmatrix} = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{pmatrix} u_a \\ u_b \end{pmatrix} \tag{3.10}$$

$$\begin{pmatrix} f_{b2} \\ f_c \end{pmatrix} = \begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{pmatrix} u_b \\ u_c \end{pmatrix} \tag{3.11}$$

(3.10) 式、(3.11) 式を足し合わせることで (3.12) 式を得ることができる。

$$\begin{pmatrix} f_a \\ f_{b1} + f_{b2} \\ f_c \end{pmatrix} = \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{pmatrix} u_a \\ u_b \\ u_c \end{pmatrix} \tag{3.12}$$

f_b は質点 B にかかる力なので、AB の接続と BC の接続を考慮した質点 B にかかる力は (3.13) 式で表せる。

$$f_b = f_{b1} + f_{b2} \tag{3.13}$$

また、複数の要素どうしで接続した場合は上記に示した足し合わせの手順を接続関係に即して行う必要がある。横方向質点数を 2、縦方向質点数を 2 とする紐帯モデルの各質点をそれぞれ **A**、**B**、**C**、**D** とする。図 3.7 は頂点をそれぞれ割り振った様子を表している。

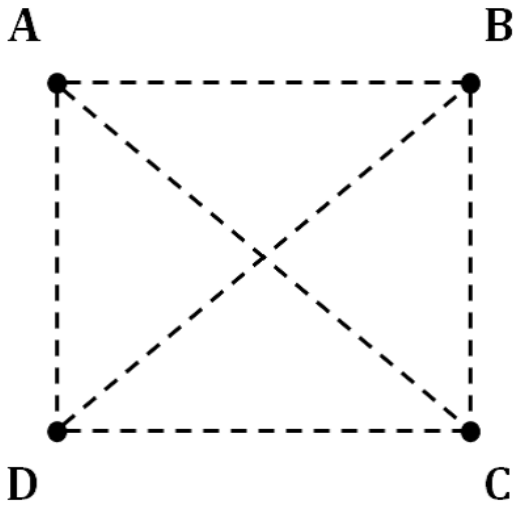

図 3.7　複数の弾性体要素どうしでの接続の例

接続 **AB**、**BC**、**CD**、**AD** については連続した接続関係なので前述と同様の足し合わせを行うことが計算できる。しかし、接続 **AC**、**BD** の接続関係を適用するには全体剛性行列を作成する際にその接続関係に即した足し合わせを行う必要がある。

質点 AC の要素行列を表 3.1 に示す。表 3.1 において、**A** 行 **A** 列要素は質点 **A** にかかる力を求めるための成分、**A** 行 **C** 列要素は **A** が **C** に及ぼす力を求めるための成分、**C** 行 **A** 列要素は **C** が **A** に及ぼ す力を求めるための成分、**C** 行 **C** 列要素は **C** にかかる力を求めるための成分である。

表 3.1　AC の要素行列

	A	**C**
A	k_{AC}	-k_{AC}
C	-k_{AC}	k_{AC}

ここで全体剛性行列の時数をあらかじめ質点数分確保し、行と列に各質点を対応したものとする。この時の四角形 ABCD における全体剛性行列を (3.14) 式に表す。ここで、行列内の各要素

は、表 3.1 における行列要素の定義と同様である。

$$
\begin{aligned}
&\begin{bmatrix} k_{AB} & -k_{AB} & 0 & 0 \\ -k_{AB} & k_{AB} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & k_{BC} & -k_{BC} & 0 \\ 0 & -k_{BC} & k_{BC} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \cdots \\
&\cdots + \begin{bmatrix} k_{AC} & 0 & -k_{AC} & 0 \\ 0 & 0 & 0 & 0 \\ -k_{AC} & 0 & k_{AC} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & k_{BD} & 0 & -k_{BD} \\ 0 & 0 & 0 & 0 \\ 0 & -k_{BD} & 0 & k_{BD} \end{bmatrix} \\
&= \begin{bmatrix} k_{AB}+k_{AC}+k_{AD} & -k_{AB} & -k_{AC} & -k_{AD} \\ -k_{AB} & k_{AB}+k_{BC}+k_{BD} & -k_{BC} & -k_{BD} \\ -k_{AC} & -k_{BC} & k_{AC}+k_{BC}+k_{CD} & -k_{CD} \\ -k_{AD} & -k_{BD} & -k_{CD} & k_{AD}+k_{BD}+k_{CD} \end{bmatrix}
\end{aligned}
\tag{3.14}
$$

(3.14) 式のように、接続関係に即した足し合わせをすることにより、複雑な接続状態を表した全体剛性行列を作成することが可能となる。

3.4.3 拘束点の設定

前節で作成した全体剛性行列に対して拘束点を設定することで、移動や変形に制限を与え、変形しない弾性体の要素の表現が可能となる。本研究ではユーザーが任意に拘束点を設定できるようにしている。拘束点は有限要素法による変位を反映しないので、他のモデルに接続して使う場合に接続する質点を拘束点に設定することで、接続した質点はモデルに対して固定される。

全体剛性行列への拘束点の設定は、全体剛性行列の大きさを n 行 n 列、拘束点を u_i とすると、行列の i 行 i 列の全成分を対角項を除いてすべて 0 に設定し、対角項を 1 にすることで実現できる。

(3.15) 式は拘束条件を全体剛性行列に適用する様子を表している。

$$\begin{bmatrix} k_{11} & \cdots & 0 & \cdots & k_{1n} \\ \cdots & \cdots & 0 & \cdots & \cdots \\ 0 & 0 & 1 & 0 & 0 \\ \cdots & \cdots & 0 & \cdots & \cdots \\ k_{n1} & \cdots & 0 & \cdots & k_{nn} \end{bmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_i \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ 0 \\ \vdots \\ f_n \end{pmatrix} \tag{3.15}$$

拘束点を設定し、全体剛性行列の逆行列を導出可能な正則行列を生成する。なお、拘束点の組み合わせや質点同士の幾何的関係から全体剛性行列が非正則となることがある。その場合は有限要素法による変形処理は行わず、後述するバネ・質点系変形処理のみを行うことで継続して変形処理を実現した。

以降、拘束条件を設定していない全体剛性行列を拘束条件を $\mathbf{M}_1$、設定した全体剛性行列を $\mathbf{M}_2$ として説明を行う。

3.4.4 有限要素法による変形処理

一般的な有限要素法による計算で扱う変位は、計算のタイムステップにおける質点の現在位置と質点の計算開始時の初期位置との差を取ることで算出する。しかし、本研究では質点の現在位置と 1 つ前のタイムステップにおける位置との差で求めた変位を扱う。

ここで、$\mathbf{U}_i(0 \leqq i \leqq n-1)$ を変位として、$\mathbf{U} = (\mathbf{U}_0, \cdots, \mathbf{U}_{n-1})^T$ を変位ベクトル行列とする。同様に、$\mathbf{P} = (\mathbf{P}_0, \cdots, \mathbf{P}_{n-1})^T$ を質点位置ベクトル 行列、$\mathbf{R} = (\mathbf{R}_0, \cdots, \mathbf{R}_{n-1})^T$ を 1 つ前のタイムステップでの質点位置ベクトルとする。

$$\mathbf{U} = \mathbf{P} - \mathbf{R} \tag{3.16}$$

(3.16) 式で求めた変位と $\mathbf{M}_1$ を (3.3) 式に当てはめることで 3.4.3 節の処理を適用する前のモデ

ル全体の力 $\mathbf{F}_1$ を求めることができる。

$$\mathbf{F}_1 = \mathbf{M}_1\mathbf{U} \tag{3.17}$$

(3.17) 式で求めた $\mathbf{F}_1$ に対して、拘束点にかかる力を 0 に設定することで拘束条件を考慮した質点の移動処理を実現する。拘束条件を適用した力を $\mathbf{F}_2$ とすると (3.17) 式より、変形後の変位は、全体剛性行列の逆行列と力とを乗ずることで求めることができる。よって $\mathbf{M}_2$ の逆行列に $\mathbf{F}_2$ を乗ずることで変形後の変位 $\mathbf{U}'$ を算出した。(3.18) 式は変形後の変位を求める式である。

$$\mathbf{U}' = \mathbf{M}_2^{-1}\mathbf{F}_2 \tag{3.18}$$

(3.18) 式で求めた変形後の変位を $\mathbf{P}$ に足し合わせることで各質点の位置の更新を行い、変形後の質点の位置 $\mathbf{P}'$ を算出して変形を実現する。(3.19) 式に変位の更新を行うための式を示す。

$$\mathbf{P}' = \mathbf{P} + \mathbf{U}' \tag{3.19}$$

3.5 バネ・質点系による変位の補正

有限要素法の適用後、バネ・質点系に基づいてさらに変位の補正をおこなう。質点 $\mathbf{P}_a$ と隣接している質点 $\mathbf{P}_i$ を接続している辺をバネ (バネ定数は k) とみなすと、フックの法則により辺の自然長 d_i からの変位をもとに $\mathbf{P}_i$ に接続された j 本の辺が自然長に戻ろうとする力 $\mathbf{U}_a$ を算出できる。

$$\mathbf{U}_a = \frac{1}{2}\sum_{i}^{j} k(d_i - |\mathbf{P}_a - \mathbf{P}_i|)(\mathbf{P}_a - \mathbf{P}_i) \tag{3.20}$$

(3.20) 式で算出したモデル全体の頂点の位相を表す行列を $\mathbf{U}'$ とし、(3.19) 式で算出した頂点 $\mathbf{P}'$ に加算することで補正後の位置 $\mathbf{P}''$ を算出する。(3.21) 式は変位の補正で求められた補正ベクトルを用いて頂点移動を行う様子を表している。

$$\mathbf{P}'' = \mathbf{P}' + \mathbf{U}' \tag{3.21}$$

3.6　自己衝突判定による変位の補正

自身の形状が変形しない剛体に対し、弾性体では自身の一部と衝突する場合がある。以下に自己衝突を行う場合と行わない場合の例を図 3.8、図 3.9 示す。

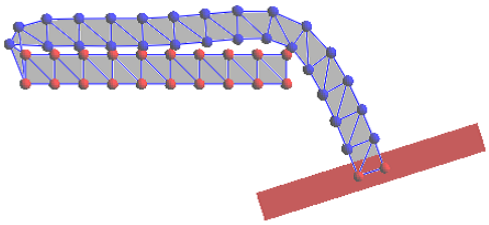

図 3.8　自己衝突判定あり

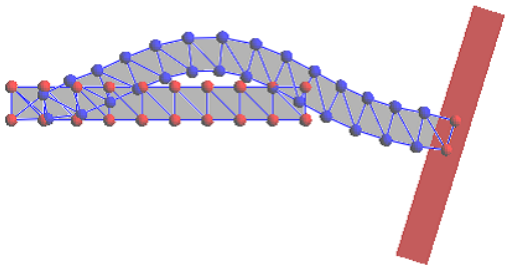

図 3.9　自己衝突判定なし

自己衝突の検出には 3.1.2 項で設定した境界ボリュームの状態をもとに境界ボリュームの交差を算出する。また、すべての質点と弾性体の要素とで衝突判定を行うと判定の数自体が膨大になってしまう。そこで、ある質点に注目したとき同じ質点を共有する三角形とは衝突判定を行わないようにする。

円形境界ボリュームは質点の座標を、三角形境界ボリュームは頂点となる 3 つの質点の座標をもとに設定する。円形境界ボリュームの基準とする質点を $\mathbf{C}_p$、三角形境界ボリュームの基準とする 3 頂点をそれぞれ $\mathbf{T}_a$、$\mathbf{T}_b$、$\mathbf{T}_c$ とすると $\mathbf{C}_p = \mathbf{T}_a$, $\mathbf{C}_p = \mathbf{T}_b$, $\mathbf{C}_p = \mathbf{T}_c$ のいずれかが成立した時点でその境界ボリュームどうしの衝突判定は無視される。図 3.10 はある質点に注目した場合衝突判定が無視される三角形の境界ボリュームを示している。

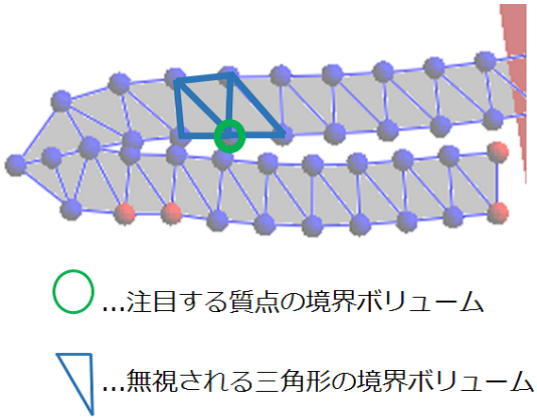

図 3.10　自己衝突判定あり

さらに、質点や弾性体の要素の境界ボリュームに対し空間四分木 [18] を適用する。これらによって衝突判定の数を少なくする。

3.6.1　自己衝突の判定

本項では自己衝突を検出するために、弾性体の要素に設定した三角形境界ボリュームと質点に設定した円形境界ボリュームとの交差判定を行う手法について述べる。

半径を r とする質点の円形境界ボリュームの中心の位置ベクトルを $\mathbf{C}$ とする。弾性体の要素の三角形境界ボリュームの重心を $\mathbf{T}$、各頂点を $\mathbf{T}_1$ $\mathbf{T}_2$ $\mathbf{T}_3$ とする。それぞれの形状の境界ボリュームを図 3.11 に示す。

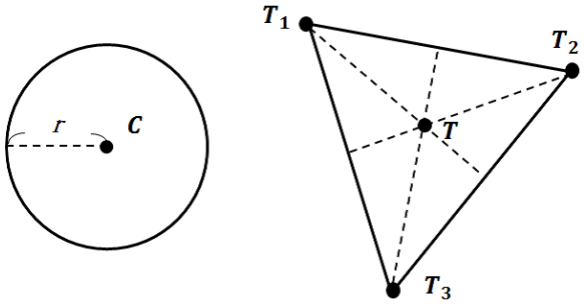

図 3.11　円と三角形の境界ボリュームの例

$\mathbf{C}$ から三角形の境界ボリュームへの最近接点を求め、その点を $\mathbf{T}_c$ とする。最近接点とは、あ

る図形に対してその点から最も近い点である。図 3.12、図 3.13 は、円形境界ボリュームが三角形境界ボリュームが交差している場合と交差していな場合を表している。

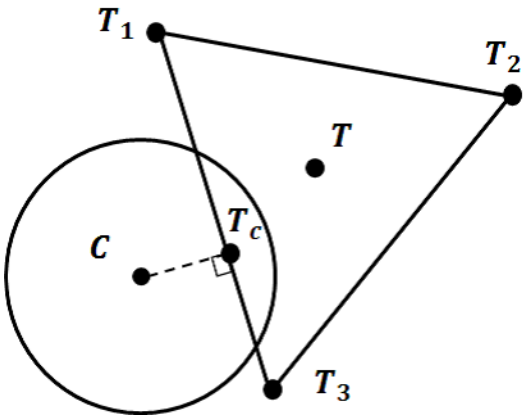

図 3.12　交差している

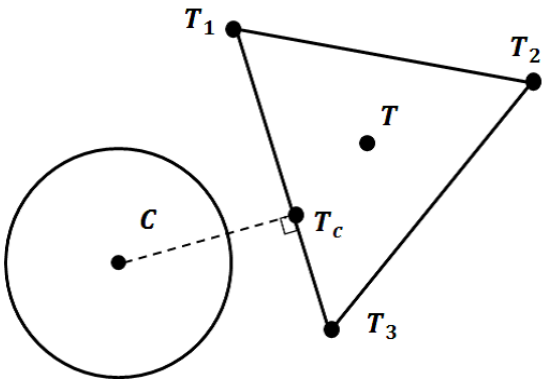

図 3.13　交差していない

円の領域は $\mathbf{C}$ から半径 r の距離以内にある点であることから、(3.22) 式を満たすとき円の境界ボリュームと三角形の境界ボリュームは交差していることになる。

$$|\mathbf{C} - \mathbf{T_c}| < r \tag{3.22}$$

以降、本研究では境界ボリュームが交差していた場合、境界ボリュームを設定したオブジェクトが衝突したものとみなす。

3.6.2　自己衝突の衝突応答

前項で衝突を検出した場合、境界ボリュームどうしの交差を解消するためのベクトルを算出する。

$\mathbf{T}_c$ は三角形の境界ボリュームの領域において $\mathbf{C}$ に最も近い点であることから、ベクトル $\overrightarrow{\mathbf{CT}_c}$ は円形境界ボリュームと三角形境界ボリュームが最も深く交差している方向を表している。ベクトル $\overrightarrow{\mathbf{CT}_c}$ の方向を表すベクトルを

$$\mathbf{d} = \frac{(\mathbf{C} - \mathbf{T}_c)}{|\mathbf{C} - \mathbf{T}_c|} \tag{3.23}$$

とすると、$\mathbf{d}$ の向きに注目すると (3.24) 式により円形境界ボリュームが三角形境界ボリュームに交差している深さ i が求められる。

$$i = |\mathbf{C} - \mathbf{T}_c| - r \tag{3.24}$$

よって、境界ボリュームの交差を解消するベクトル $\mathbf{n}$ は (3.25) 式より求められる。

$$\mathbf{n} = i\mathbf{d} \tag{3.25}$$

図 3.14 は (3.24) 式、(3.25) 式における各ベクトルとノルムの関係を表している。

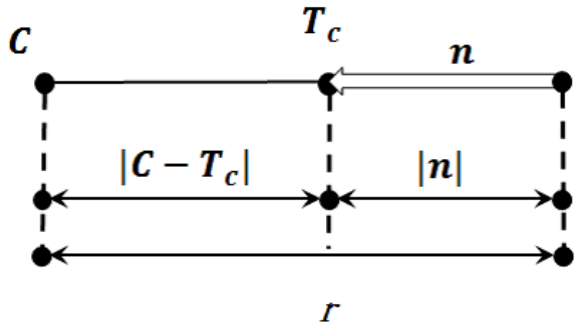

図 3.14　交差を解消するベクトル n の算出方法

交差を解消するベクトル $\mathbf{n}$ を、3.1.2 節で設定した m_c と m_t の値を (3.26) 式、(3.27) 式にあてはめて円形の境界ボリュームの交差解消ベクトル $\mathbf{n}_c$ と三角形の境界ボリュームの交差解消ベクトル $\mathbf{n}_t$ に分解し、それぞれ適用して交差を解消する。

$$\mathbf{n}_c = \frac{m_\mathbf{c}}{m_\mathbf{c} + m_\mathbf{t}}\mathbf{n} \tag{3.26}$$

$$\mathbf{n}_t = -\frac{m_\mathbf{t}}{m_\mathbf{c} + m_\mathbf{t}}\mathbf{n} \tag{3.27}$$

(3.26) 式、(3.27) 式のベクトルとノルムの関係を図 3.15 に示す。

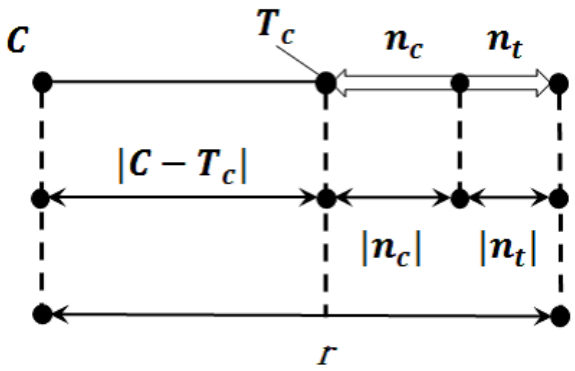

図 3.15　ベクトル n の分解

3.7　他のオブジェクトとの衝突による変位補正

弾性体と複雑な形状の境界ボリューム他の物体が衝突判定を行う場合は、弾性体の形状を単純な形状の境界ボリュームの集合で近似するツリーリフィッティング [19] などを用いて計算を簡略化しなければ実用的な運用が望めない。本研究では他の単純な形状のオブジェクトとして円形の境界ボリュームを持つ剛体モデルとの衝突処理による形状変形を行う。

まず剛体モデルと衝突する可能性がある紐帯モデルの弾性体要素を空間分割で概算し、そして衝突する可能性がある剛体モデルの境界ボリュームと弾性体要素の境界ボリュームに対して 3.6.1 項や 3.6.2 項と同様の手法を用いることで他のオブジェクトとの衝突判定と交差解消を実現した。図 3.16 は紐帯モデルが他のオブジェクトと衝突している状態である。

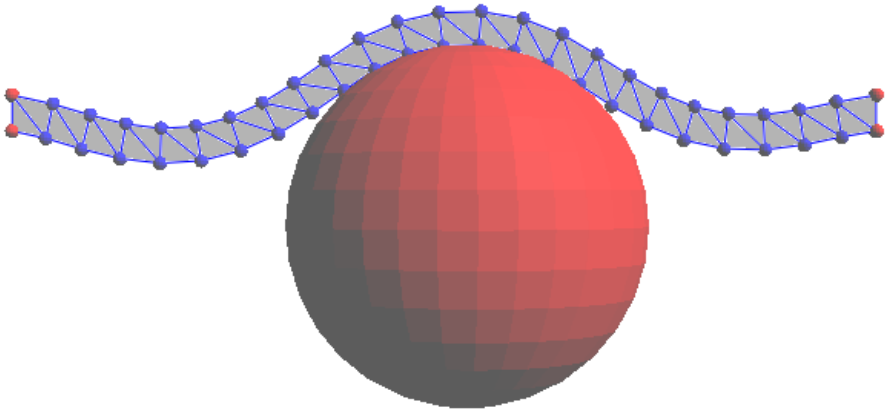

図 3.16　剛体と紐帯モデルの衝突

第 4 章

検証と評価

本章では、本研究で提案した紐帯モデルを用いて実装したアプリケーションを使用し、その有用性を検証する。本研究で試作したアプリケーションはグラフィクス API の OpenGL[20] をベースとした 3 次元グラフィクスツールキットである Fine Kirnel Toolkit[21] を用いて実装した。

4.1 実行結果

本研究で提案した手法を実装したアプリケーションの実行結果を以下に示す。本研究の処理速度検証に用い多コンピュータの環境は表 4.1 のとおりである。

表 4.1 アプリケーションの実行環境

CPU	Intel Core i7-4770 CPU @3.40GHz
RAM	16.0GB
GPU	Intel(R) HD Graphics4600

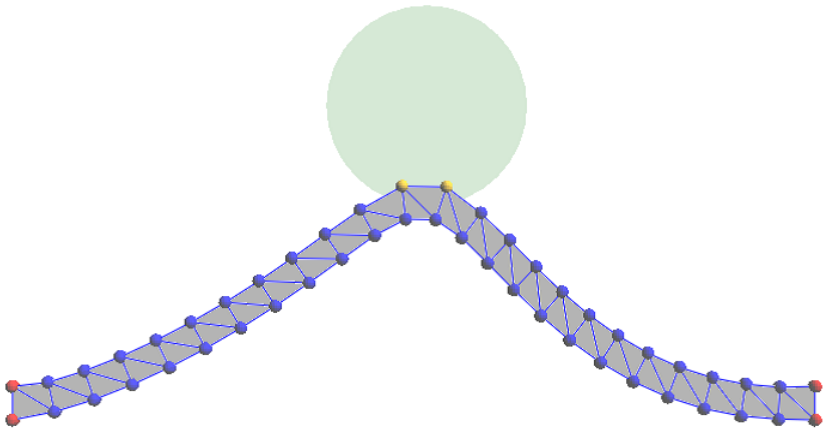

図 4.1　ユーザーの入力による変形.

図 4.1 は提案手法を用いた弾性体モデルの変形挙動を示している。図中の青い点が弾性体の質点、図中の黄色い点が操作している質点、図中の赤い点が拘束点を表している。

4.2　実行結果の比較と速度検証

本研究で提案した手法を実装したアプリケーションにおいて、質点数や各種パラメータなどの条件を変えつつ処理速度を検証した。それぞれの条件の設定内容を表 4.2 に示す。

表 4.2　本研究で制作したシステムの実行結果

条件	設定内容
A	横方向に質点を多数の配置した場合
B	横方向と縦方向に複数の質点を配置した場合
C	条件 A より横方向に細かく分割して質点を配置した場合
D	条件 C より反復計算回数を減らして同じ分割数のモデルを用いた場合

各条件におけるパラメータと処理速度を表 4.3 に示す。処理速度の単位は FPS(Frame Per Second) であり、これは 1 秒間に処理可能なシミュレーションステップ数を表す。また、各条件でシミュレーションを実行した際のスクリーンショットをそれぞれ図 4.2、図 4.3、図 4.4 に示す。

表 4.3　本研究で制作したシステムの実行結果

条件	総質点数	バネ係数	反復計算回数	実行速度 (fps)
A	50(横:25, 縦:2)	0.45	8	72.1
B	50(横:10, 縦:5)	0.45	8	70.5
C	70(横:35, 縦:2)	0.45	8	32.7
D	70(横:35, 縦:2)	0.45	4	62.5

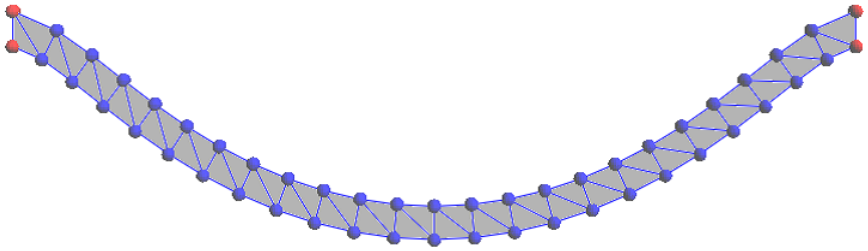

図 4.2　条件 A の実行結果

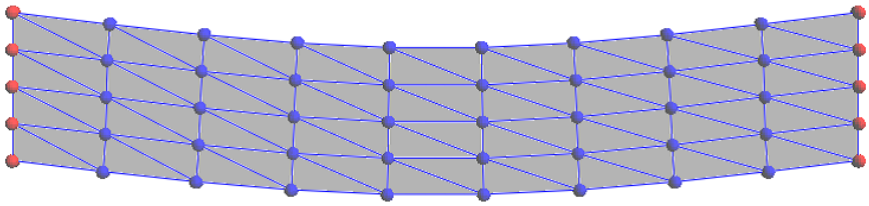

図 4.3　条件 B の実行結果

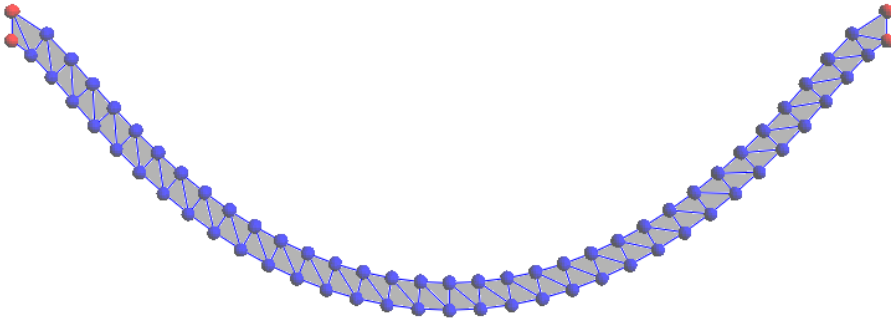

図 4.4　条件 C、D の実行結果

検証の結果、質点数や反復計算回数が増加するとともに処理速度が減少していることが分かった。また、図 4.4 における実行結果ついても反復計算回数を減らすことで実行速度を保てること

が分かった。

反復計算回数を減らすと、紐帯の変形計算の精度が落ちたり、自己衝突判定で衝突を検出しづらくなる。また、本研究ではゲーム等のインタラクティブコンテンツにおいてリアルタイム計算可能であることを目的としているが、利用する環境では紐帯と衝突させたい他のオブジェクトの数や、求められる紐帯の計算精度が異なる。よって、利用する環境に応じて各々パラメータの調整が必要である。

4.3　現状の問題点

現状の問題点として、2.2 節で、紐帯は繊維を束ねた複合構造を取っているが本手法で提案したモデルではそれを考慮しておらず、細長い弾性体であるといえる。

また、図のように紐帯が局所的に変形した場合、弾性体要素がつぶれてしまい三角形の面積が 0 を超えて負の値になってしまう。これは 3.6 節で示したように、質点を共有している弾性体要素の境界ボリュームとは衝突判定をとっておらず、質点の接続に関して角度制限を設けていないためである。局所的変形により弾性体の要素の面積が 0 を超えて負の値になってしまった様子を図 4.5 に示す。

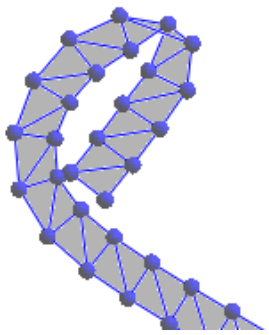

図 4.5　局所的変形により要素がつぶれてしまっている

第 5 章

終わりに

本研究ではゲームなどのリアルタイム計算とインタラクティブ性が求められる状況での紐帯のシミュレーション手法を提案した。有限要素法とバネ・質点系により紐帯モデルの変形、紐帯モデルの自己衝突判定、紐帯モデル以外の物体との衝突をリアルタイム計算で実現した。しかし、有限要素法や自己衝突判定に計算コストがかかるため、紐帯モデルの質点数の増加によって処理速度の低下が起こることを確認した。

今後はさらに質点や反復計算回数を増やしたうえで処理速度を維持しするために、処理のボトルネックになっている有限要素法の計算をに行うために、逆行列演算を高速に行うための対応が必要になる。そのうえで 3 次元弾性体でも本手法を用いたシミュレーションを行い、その有用性を検証していきたい。

また、4.3 節で述べたように紐帯の局所変化時に弾性体要素がつぶれてしまい三角形の面積が負の値になってしまうので、角度バネや弾性体要素の三角形面積による補正処理などを適用する必要がある。

さらに、2.3 節で述べたように剛性を踏まえて変形計算をすることで紐帯の伸長度を表現したが、繊維を束ねることで生じる性質は表現できていないので、リアルタイム計算を念頭において

繊維構造を簡易的に計算する対応が必要である。

なお、本研究は NICOGRAPH2016 ポスター発表において“紐帯のリアルタイム力学シミュレーションに関する研究”[22] として発表した内容を含む。

謝辞

本研究を締めくくるにあたり、手厚くご指導を頂いた渡辺大地講師と阿部雅樹様に感謝の意を表します。また、締め切りが迫るたびに研究生活を共にした研究室のメンバーにも御礼を申し上げます。

参考文献

[1] CGWORLD. *CGWORLD + digitalvideo vol.206*. ボーンデジタル, 2015.

[2] Bullet Physics Library. Real-time physics simulation. http://bulletphysics.org/wordpress/. 参照: 2016-8-7.

[3] 広田光一, 金子豊久. 仮想物体の弾性モデルに関する検討. 計測自動車制御学会論文誌 34(8), Vol. 34, No. 3, pp. 232–238, 1998.

[4] 小山裕己, 高山健志, 梅谷信行, 五十嵐健夫. 例示ベースの弾性変形の実時間計算手法. 情報処理学会論文誌, Vol. 54, No. 10, pp. 232–238, 2013.

[5] 永安伸大, 近野敦, 辻田哲平, 佐瀬一弥, 小水内俊介. 縫合シミュレーションのための糸のモデリングおよび生体組織との接触判定. 第 32 回日本ロボット学会学術講演会, 2014.

[6] 中島佳衣, 渡辺大地. 物体の内部組織構造を考慮した力覚表現に関する研究. Nicograph 論文コンテスト論文集, 東京工科大学メディア学部ゲームサイエンスプロジェクト, 2009.

[7] W.H.Press, B.P.Flannery, S.A.Teukolsky, W.T.Vetterling. Numerical Recipes in C : C 言語による数値計算のレシピ. 技術評論社, 1993.

[8] 越塚誠一. 粒子法による流れの数値解析. ながれ, Vol. 21, pp. 230–239, 2002.

[9] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical*

Journal, Vol. 82, pp. 1013–1024, 1977.

[10] Matthias Muller, Bruno Heidelberger, Bruno Heidelberger, and Bruno Heidelberger. Meshless deformations based on shape matching. *ACM Transactions on Graphics*, Vol. 24, No. 3, pp. 471–478, 2005.

[11] 海上一徳, 渡辺大地. リアルタイム 3dcg における位相変化を考慮した弾性体の挙動表現に関する研究. 情報処理学会 第 71 回全国大会, 東京工科大学メディア学部ゲームサイエンスプロジェクト, 2008.

[12] 東京製綱. ワイヤロープ(ワイヤーロープ)の概要. https://www.tokyorope.co.jp/product/wirerope/outline.html. 参照: 2016-12-18.

[13] 株式会社フジックス. 縫い糸の種類. http://www.fjx.co.jp/learn/knowledge01.html. 参照: 2016-12-18.

[14] Jonathan M. Kaldor, Doug L. James, and Steve Marschner. Simulating knitted cloth at the yarn level. *SIGGRAPH*, Vol. 27, No. 65, 2008.

[15] Christer Ericson. Real-Time Collision Detection : ゲームプログラミングのためのリアルタイム衝突判定. ボーンデジタル, 2005.

[16] ジェイソン・グレゴリー. ゲームエンジン・アーキテクチャ. ソフトバンククリエイティブ, 2010.

[17] 恒藤敏彦. 弾性体と流体. 岩波書店, 1983.

[18] Gabriel Zchmann Elmar Langetepe. 空間的データ構造とアルゴリズム. 株式会社ボーンデジタル, 2007.

[19] 松生 裕史原田 隆宏. ゲーム制作者のための物理シミュレーション入門　剛体編. インプレスジャパン, 2012.

[20] OpenGL.org. Opengl. `http://www.opengl.org/`. 参照: 2016-8-8.

[21] Fine Kernel Project. Fine kernel toolkit system. http://fktoolkit.sourceforge.jp/. 参照: 2016-8-8.

[22] 後藤佳樹, 渡辺大地. 紐帯のリアルタイム力学シミュレーションに関する研究. NICOGRAPH2016 ポスター発表, pp. 137–138, 2016.