コンピュータアニメーション特論

第6回　物理シミュレーション

九州工業大学 情報工学研究院　尾下真樹

今日の内容

- 物理シミュレーションの概要
- 剛体の物理シミュレーション
  - 運動方程式
  - 回転運動と慣性モーメント
  - シミュレーションの手順
- 衝突と接触の扱い
- 多関節体の物理シミュレーション
- 変形する物体の物理シミュレーション

物理シミュレーション

- 物理シミュレーション（Physics Simulation）
  = 動力学シミュレーション（Dynamics Simulation）
  - 物理法則に従って物体の運動を計算
    - 物理的に正しいアニメーションが生成できる
  - 物体の位置を直接動かすのではなく、力を加えることで間接的に物体が運動する
    - 意図した通りにコントロールすることは難しい
- 物理シミュレーションは、さまざまな用途で用いられる

今日の内容

- 物理シミュレーションの概要
- 剛体の物理シミュレーション
  - 運動方程式
  - 回転運動と慣性モーメント
  - シミュレーションの手順
- 衝突と接触の扱い
- 多関節体の物理シミュレーション
- 変形する物体の物理シミュレーション

参考書

- 「3DCGアニメーション」
  栗原恒弥・安生健一 著、技術評論社、¥2,980
  - アニメーション技術全般を解説
- ロボット工学 - 機械システムのベクトル解析
  広瀬 茂男著、装華房、¥4,950
  - 剛体・多関節の動力学計算

物理シミュレーションの概要

物理シミュレーションの応用例（1）

- シミュレータ
  - ロボットや工業製品の評価・設計などに利用
  - なるべく正確なシミュレーションが要求される
  - 必要であれば計算時間がかかっても良い
- アニメーション
  - 映画などへの利用、物理現象による動きの生成
  - 見た目が自然であることが重要（必ずしも物理的に正確でなくても構わない）
  - 結果をコントロールしやすいことも要求される

物理シミュレーションの応用例（2）

- コンピュータゲーム
  - アニメーションと同様、見た目が自然で、コントロールしやすいことが重要
  - 高速に計算できることが求められる
- 最近は物理シミュレーションを取り入れたゲームも多く存在
  - カーレース、フライトシミュレータ、弾道計算など
  - 自然な動きを生成しようとすると、物理法則を取り入れるのが最も有効
  - 必要に応じてパラメタやモデルを調整

物理シミュレーションの応用例（3）

- ゲームにおける物理シミュレーションの目的
  - ゲームプレイ物理（Game Play Physics）
    - 物理計算を使って乗り物や人物の動きを生成するなど、物理計算がゲーム進行に影響を与えるもの
    - ゲームの難易度等にも影響するため、必ずしも物理的に正確な動きを表現するだけでなく、制作者の意図通りの動きを実現することが要求されることがある
  - 効果物理（Effects Physics）
    - 炎・煙・水面の表現、爆発の効果の表現、やられて落ちる人物の表現など、ゲーム進行には影響しないが、映像のリアリティを増すために物理計算を用いるもの
    - 比較的容易に導入しやすい

物理シミュレーションの実現方法

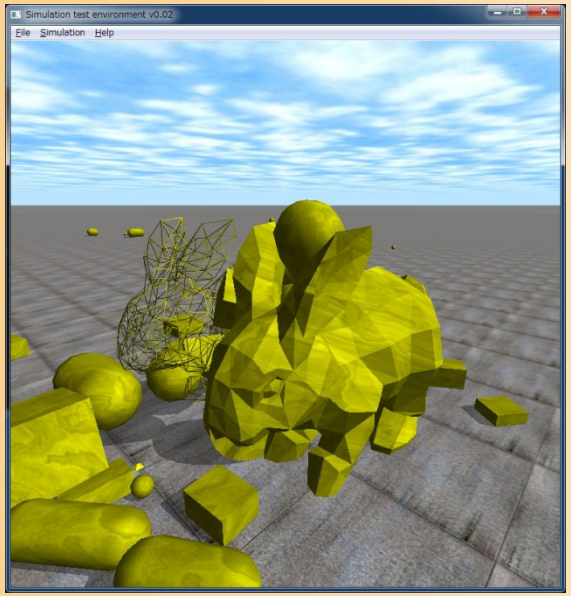
- 汎用のライブラリを使用
  - 最近は、汎用の物理シミュレーションライブラリ（ミドルウェア）が広く利用されている
    - ODE, Bullet, Havok, PhysX, OpenHRP等
      - 市販のコンピュータゲームなどでも使用されている
    - Unity, Unreal 等のゲーム用ミドルウェアも物理演算機能を持つ
    - インタラクティブなアニメーション生成が目的であり、精度は高くない
    - ある程度物理シミュレーションの原理を理解していなければ、使えない
- 自分で開発
  - 基本的な物理シミュレーションの原理や実装はそれほど難しくない
  - 安定性や高速化を実現しようとすると工夫が必要
  - 物理シミュレーションの手法自体に工夫をしたい場合などは、基本的に自作する必要がある

物理シミュレーション ミドルウェア

- ODE（Open Dynamics Engine）
  - フリーで利用可能
  - 剛体（多関節体）の運動シミュレーション
- Bullet
- Havok（Havok社 → Intelが買収）
  - 剛体（多関節体）や粒子の運動シミュレーション・描画
  - グラフィックカード（GPU）を使って高速に計算可能
- PhysX（AGEIA社 → nVidiaが買収）
  - もともとは、専用プロセッサ（カード）を使用
  - グラフィックカード（GPU）でも動作するようになった
- OpenHRP（日本で開発されたロボットシミュレータ）

物理シミュレーション ミドルウェアの例

- Open Dynamic Engine (ODE)
  - http://www.ode.org
    - ライブラリ＋サンプルのソースが公開

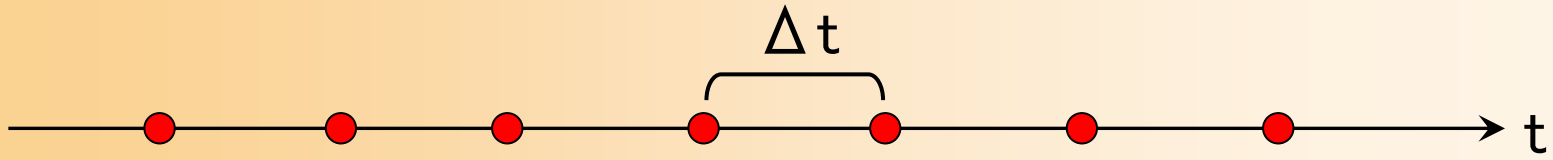


参考書:
簡単!実践! ロボットシミュレーション - Open Dynamics Engineによる ロボットプログラミング
出村公成、森北出版、3,360円

Simulation test environment v0.02
File
Simulation
Help
物理シミュレーション
Pysics Simulation (ODE)

物理シミュレーションの種類

- 解析的なシミュレーション
  - 運動中に外力などが加わらず、運動の軌道が既知の場合（数式などによって表される場合）は、数式にもとづいて運動を計算することが可能
- 数値計算的なシュミレーション
  - 運動中に外力などが加わる場合
  - 解析的には解けないため、各離散時間ステップごとに、外力などを考慮して運動を計算
  - 精度の問題



対象となる物体の種類

- 剛体（Rigid Body）
  - ニュートンの運動方程式に従う
  - 質点のみ考えれば良いので比較的簡単
- 多関節体（Articulated Body）
  - ロボット・人体など
- 変形する物体（Deformable Object）
  - 布、クッション、衣服、皮膚、髪の毛など
  - 粒子モデルや有限要素法などを用いて形状変形を計算
- 不定形の物体
  - 炎、雲、水面など
  - 粒子モデル、ボクセルモデルなどを用いて分布を計算

対象となる物体の種類

- 剛体（Rigid Body
  - ニュートンの運動方
  - 質点のみ考えれば良

対象となる物体の種類に応じて、異なる物理シミュレーション手法やモデルが必要となる

- 多関節体（Articulated Body）
  - ロボット・人体など
- 変形する物体（Deformable Object）
  - 布、クッション、衣服、皮膚、髪の毛など
  - 粒子モデルや有限要素法などを用いて形状変形を計算
- 不定形の物体
  - 炎、雲、水面など
  - 粒子モデル、ボクセルモデルなどを用いて分布を計算

剛体の物理シミュレーション

剛体の運動

- 基本的には力学の講義で習った内容の復習
  - 知識としては大学１・２年生レベル
  - 物理シミュレーションのプログラミングを行うために必要な内容を説明
- 剛体の物理シミュレーション
  - 運動方程式
  - 回転と慣性モーメント
  - 接触と衝突

運動方程式

- ニュートン・オイラーの運動方程式
  - 並進運動（ニュートンの運動方程式）

運動量

$$\mathbf{p} = m\mathbf{v}$$

質量・並進速度

運動方程式

$$\mathbf{F} = m\mathbf{a}$$

力　並進加速度

  - 回転運動（オイラーの運動方程式）

運動量

$$\mathbf{L} = \mathbf{I}\boldsymbol{\omega}$$

慣性モーメント行列・回転速度

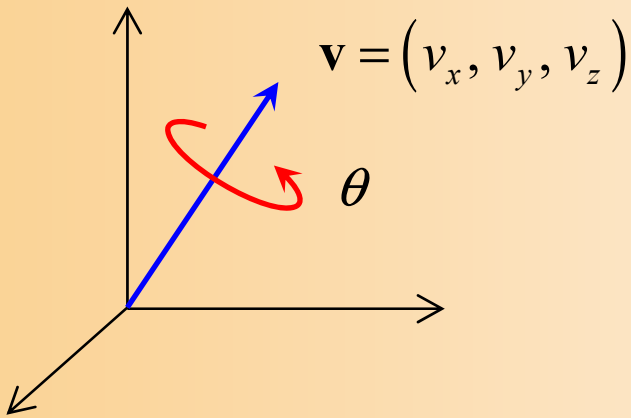
運動方程式

$$\mathbf{N} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}$$

トルク　回転加速度

回転の表現方法

- ３次元ベクトルによる回転の表現
  - 前回の講義で出てきた、回転軸と回転角度による表現と同じ
  - ベクトルの大きさが回転角度（速度・加速度）を表す



$$\mathbf{r} = \theta\mathbf{v} = (\theta v_x, \theta v_y, \theta v_z)$$

回転角度 $\theta = |\mathbf{r}|$ 回転軸 $\mathbf{v} = \dfrac{\mathbf{r}}{|\mathbf{r}|}$

回転表現の変換（復習）

- 四元数から回転行列への変換
  - 任意ベクトル周りの回転行列に相当

If the scalar part has value w, and the vector part values x, y, and z, the
worked out to be

$$M = \begin{bmatrix} 1-2y^2-2z^2 & 2 \\ 2xy-2wz & 1- \\ 2xz+2wy & 2yz-2wx \quad 1-2x^2-2y^2 \end{bmatrix}$$

when the magnitude $w^2+x^2+y^2+z^2$ equals 1. The

前回の授業で学習した方法で、回転行列への変換が可能
ただし、回転角度・速度・加速度の大きさが 180（π） を超える場合は、回転行列への変換はできない

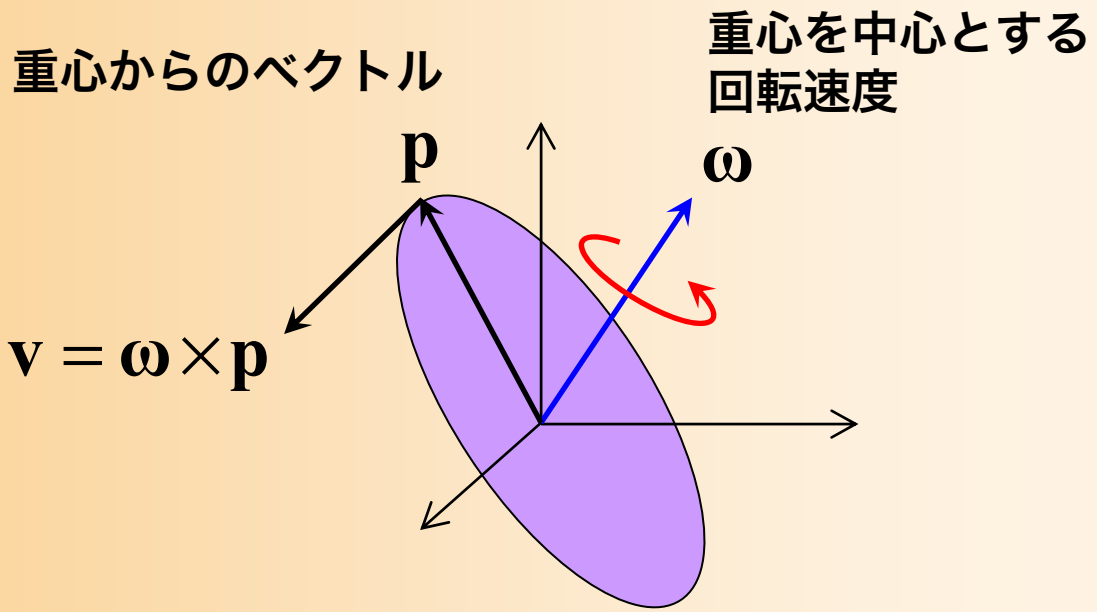
Ken Shoemake, “Animating Rotation with Quaternion Curves”, Proc. of SIGGRAPH ‘85, pp. 245-254, 1985. より

回転の表現方法の注意

- 回転角度が０になると、回転ベクトルも０となり、回転軸の情報が消失してしまう
  - 回転角度が０のときには、回転の情報には意味がないため、回転軸は任意の向きで構わない
  - プログラム的には、回転軸を求めるために正規化を行おうとすると、ゼロ割が生じて（無限大になって）しまい、問題が生じる
  - そのため、例えば、回転角度（ベクトルの長さ）が０のときは、正規化は行わず、適当な回転軸を設定する、といった対応を行う

並進運動と回転運動

- 並進運動と回転運動の関係
  - 回転速度 ω から、外積計算により、任意の点の並進速度 v を計算できる
  - 力や運動量に関しても同様



慣性モーメント行列

- 慣性モーメント行列
  - 質量の回転版
  - 回転速度に応じてどれだけの回転運動量を持つかを表す
  - ３×３の対称行列になる

運動量

$$\mathbf{L} = \mathbf{I}\boldsymbol{\omega} \qquad \mathbf{I} = \begin{pmatrix} \mathbf{I}_{xx} & -\mathbf{I}_{xy} & -\mathbf{I}_{xz} \\ -\mathbf{I}_{xy} & \mathbf{I}_{yy} & -\mathbf{I}_{yz} \\ -\mathbf{I}_{xz} & -\mathbf{I}_{yz} & \mathbf{I}_{zz} \end{pmatrix}$$

$$\mathbf{p} = m\mathbf{v}$$

慣性モーメント行列の計算

- 次の体積分により計算できる

$$\mathbf{I} = \begin{pmatrix} \mathbf{I}_{xx} & -\mathbf{I}_{xy} & -\mathbf{I}_{xz} \\ -\mathbf{I}_{xy} & \mathbf{I}_{yy} & -\mathbf{I}_{yz} \\ -\mathbf{I}_{xz} & -\mathbf{I}_{yz} & \mathbf{I}_{zz} \end{pmatrix} = \begin{pmatrix} \iiint (y^2+z^2)\,dxdydz & -\iiint (xy)\,dxdydz & -\iiint (xz)\,dxdydz \\ -\iiint (xy)\,dxdydz & \iiint (x^2+z^2)\,dxdydz & -\iiint (yz)\,dxdydz \\ -\iiint (xz)\,dxdydz & -\iiint (yz)\,dxdydz & \iiint (x^2+y^2)\,dxdydz \end{pmatrix}$$

  - 基本的な形状に関しては数学的に解ける
  - 任意形状の体積分はやや複雑になる

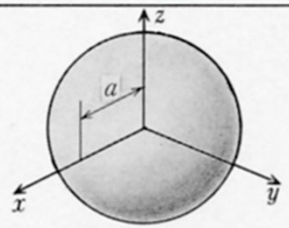
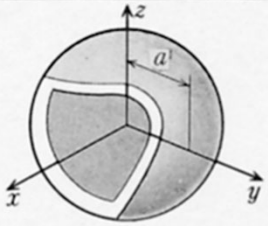
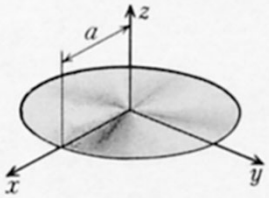
- ワールド座標系での慣性モーメント行列

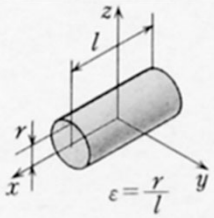
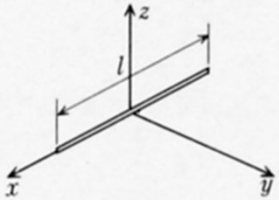
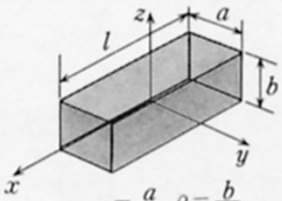
$$\mathbf{I}' = \mathbf{R}\mathbf{I}\mathbf{R}^T$$

物体の回転行列 R により計算

基本的な形状の慣性モーメント

- 球、球殻、円板、円柱、棒、直方体

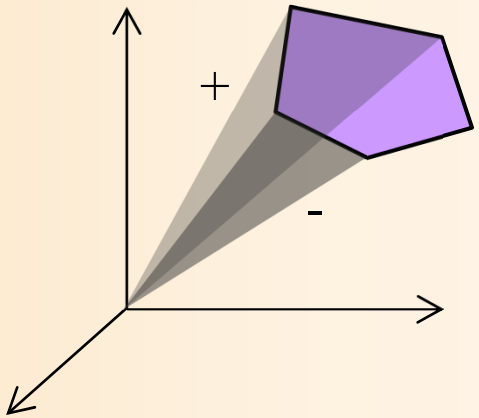
球		$I=\frac{2}{5}ma^2\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$
球　殻		$I=\frac{2}{3}ma^2\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$ (球殻の厚さはゼロとなる)
薄い円板		$I=\frac{1}{4}ma^2\begin{bmatrix}1&0&0\\0&1&0\\0&0&2\end{bmatrix}$

円　柱	 $\varepsilon=\frac{r}{l}$	$I=\frac{1}{12}ml^2\begin{bmatrix}6\varepsilon^2&0&0\\0&1+3\varepsilon^2&0\\0&0&1+3\varepsilon^2\end{bmatrix}$
細 い 棒		$I=\frac{1}{12}ml^2\begin{bmatrix}0&0&0\\0&1&0\\0&0&1\end{bmatrix}$
直 方 体	 $\alpha=\frac{a}{l},\ \beta=\frac{b}{l}$	$I=\frac{1}{12}ml^2\begin{bmatrix}\alpha^2+\beta^2&0&0\\0&1+\beta^2&0\\0&0&1+\alpha^2\end{bmatrix}$

広瀬茂男, “ロボット工学”, 装華房, p. 62 より

任意の形状の慣性モーメント

- ポリゴンモデルの慣性モーメントの計算方法
  - Sheue-ling Lien and James T. Kajiya, "A Symbolic Method for Calculationg the Integral Properties of Arbitrary Nonconvex Polyhedra“, *IEEE Cimputer Graphics & Applications*, Octover 1984, pp.35-41
  - 各ポリゴンごとに、三角すいとして体積分し、面の表裏に応じて加算・減算していく



運動の生成

- 物体は、外部から力が加えられなければ、等速運動を続ける
- 物体を運動させるためには、運動に必要な力を計算して、力を加える必要がある
  - 例：飛行機であれば、プロペラやジェットなどの推進力を計算　（エンジンのモデルが必要？）
- 衝突や接触による外部からの影響も重要
  - 何も処理を行わなければ、物体同士がめり込んでしまい、非常に不自然に見えてしまう
  - 衝突や接触の処理方法については、後述

剛体の運動のシミュレーション

- ある時刻において物体に加わる力・トルクから並進・回転加速度が計算される

$$\mathbf{a} = \frac{1}{m}\mathbf{F} \qquad \dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\left(\mathbf{N} - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}\right)$$

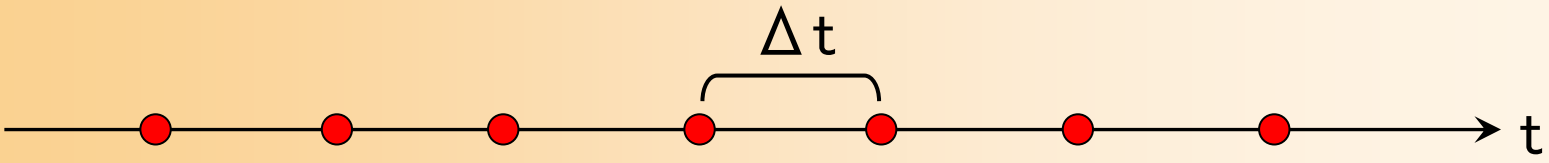
- 加速度を積分することで微小時刻における運動（位置・向きの変化）が計算できる（ニュートン法）

並進速度 $\mathbf{v}' = \mathbf{v} + \mathbf{a}\Delta t$　　回転速度 $\boldsymbol{\omega}' = \boldsymbol{\omega} + \dot{\boldsymbol{\omega}}\Delta t$

位置 $\mathbf{p}' = \mathbf{p} + \mathbf{v}'\Delta t$　　向き $\mathbf{R}' = \mathbf{M}\left(\Delta t\boldsymbol{\omega}\right)\mathbf{R}$

コンピュータによるシミュレーション

- コンピュータでシミュレーションを行う場合は、離散時間でのシミュレーションになる
  - 適当な時間間隔 Δｔ ごとに計算
    - 物体に加わる力を決め、力に応じた位置・速度変化を計算
  - Δｔ は固定の場合と、動的に変化させる場合がある（次の２種類の目的がある）
    - 精度が必要な状況で、Δｔ を小さくする
    - 速度が必要な状況で、Δｔ を大きくする（計算回数を減らす）



シミュレーションの手順

- 時間間隔 Δｔ ごとに、以下を繰り返し

1. ｔ＝ ｔ ＋Δｔ
2. 現在の時刻に各剛体に働く力・トルクを求める

   $\mathbf{F}, \mathbf{N}$
3. 力・トルクより、各剛体の位置・向きを更新

   $\mathbf{F}, \mathbf{N} \rightarrow \mathbf{v}, \mathbf{p}, \boldsymbol{\omega}, \mathbf{R}$
4. 物体同士の衝突による速度変化を計算

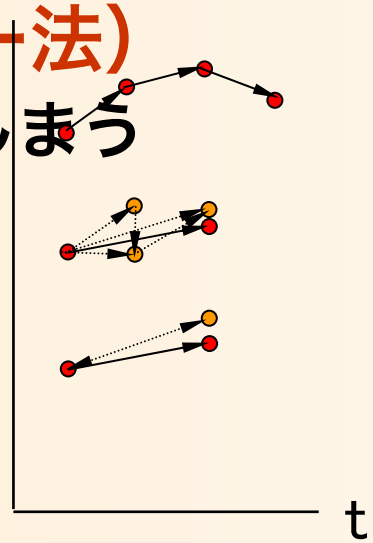
   $\mathbf{v}, \boldsymbol{\omega} \rightarrow \mathbf{v}', \boldsymbol{\omega}'$
5. 物体同士の接触を処理

   $\mathbf{v}, \mathbf{p}, \boldsymbol{\omega}, \mathbf{R} \rightarrow \mathbf{v}', \mathbf{p}', \boldsymbol{\omega}', \mathbf{R}'$

※ 詳細は、数値積分の方法や衝突・接触の処理方法によって変わる

シミュレーションの数値積分手法

- ニュートン法（ニュートン・オイラー法）
  - Δｔ が大きくなると、すぐに発散してしまう
- ルンゲ・クッタ法
  - 各ステップにおいて、中間点での位置・速度を計算し、計算結果を補正する方法
  - 1回のステップに4回の積分計算が必要
- 後退オイラー法
  - ステップの計算後の状態から、時間を逆に戻して、計算結果を補正する方法



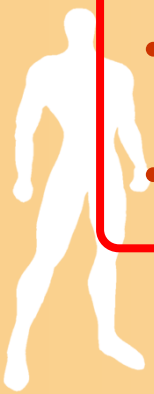
  - 下の２つの方法も、発散しにくくなくなるだけで、計算結果が正確になる訳ではないことに注意

シミュレーションの速度

- リアルタイム・シミュレーション
  - 現実世界と同じ速度でシミュレーションが進行
  - 秒間10フレーム以上（30フレーム以上が理想）
- インタラクティブ・シミュレーション
  - 対話的に操作できる程度の速度で動作
  - 必ずしも、現実世界の時間とは一致しない
  - 最低秒間数フレーム〜10フレーム以上
- オフライン・シミュレーション
  - 計算にかなりの時間がかかる
  - １フレーム数秒〜数時間

今日の内容

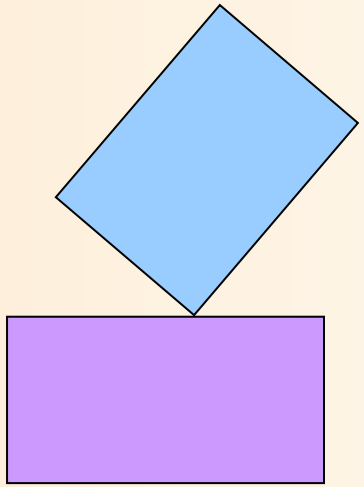
- 物理シミュレーションの概要
- 剛体の物理シミュレーション
  - 運動方程式
  - 回転運動と慣性モーメント
  - シミュレーションの手順
- 衝突と接触の扱い
- 多関節体の物理シミュレーション
- 変形する物体の物理シミュレーション



衝突と接触の扱い

衝突と接触

- 衝突と接触の２つを区別して扱うのが一般的
- 衝突（ごく短時間の接触）
  - 物体同士が初めて接触
  - 運動量保存の法則により、衝突による速度変化を計算
- 接触
  - 物体同士が継続的に接触
  - 物体同士がめり込まないように、位置・速度・加速度・力の変化を計算



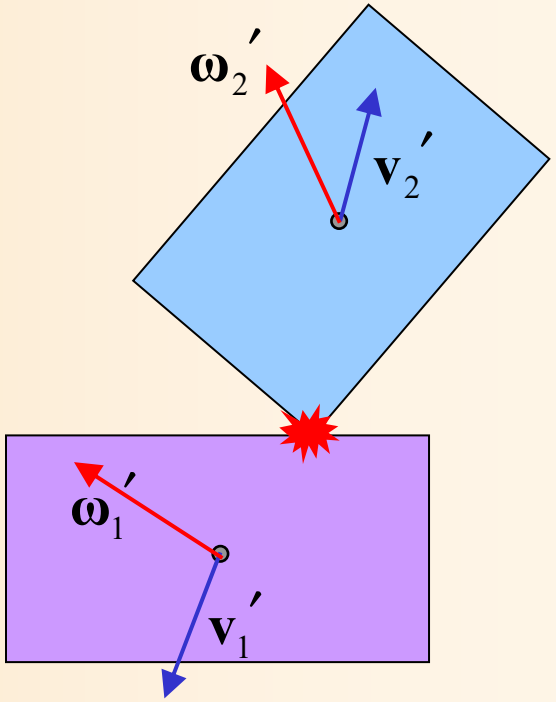
衝突の計算（1）

- 物体同士が衝突した後の、各物体の速度（並進速度・回転速度）を計算
  - 並進速度

$$\mathbf{v}_1 \rightarrow \mathbf{v}_1' \qquad \mathbf{v}_2 \rightarrow \mathbf{v}_2'$$

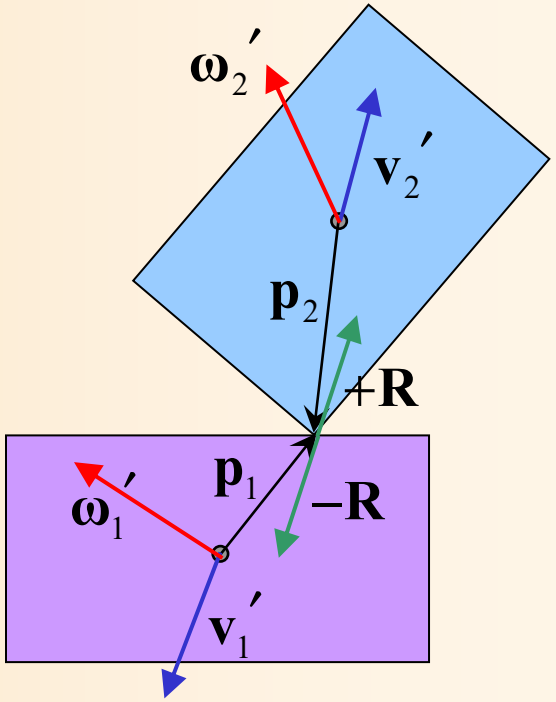
  - 回転速度

$$\boldsymbol{\omega}_1 \rightarrow \boldsymbol{\omega}_1' \qquad \boldsymbol{\omega}_2 \rightarrow \boldsymbol{\omega}_2'$$



衝突の計算（2）

- 衝突点での衝撃 R と、衝突後の両物体の速度・回転速度の方程式を立てる（続き）
  - 未知数は、5ベクトル×3次元=15
    - 解くためには15個の式が必要
  - 参考文献：
    Matthew Moore, Jane Wilhelms,
    “Collision Detection and Response for Computer Animation”,
    Computer Graphics (SIGGRAPH ’88),
    Vol. 22, No. 4, 1988.



衝突の計算（3）

- 衝突点での衝撃 R と、衝突後の両物体の速度・回転速度の方程式を立てる

$$m_1\mathbf{v}_1' = m_1\mathbf{v}_1 + \mathbf{R}$$

$$m_2\mathbf{v}_2' = m_2\mathbf{v}_2 - \mathbf{R}$$

衝突後の並進速度
（衝撃による速度変化）

$$\mathbf{I}_1\boldsymbol{\omega}_1' = \mathbf{I}_1\boldsymbol{\omega}_1 + \mathbf{p}_1 \times \mathbf{R}$$

$$\mathbf{I}_2\boldsymbol{\omega}_2' = \mathbf{I}_2\boldsymbol{\omega}_2 - \mathbf{p}_2 \times \mathbf{R}$$

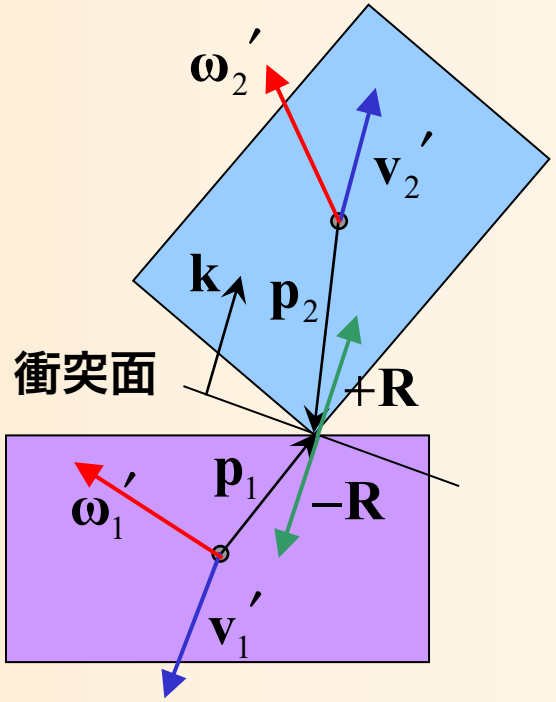
衝突後の回転速度
（衝撃による速度変化）

$$\mathbf{R}\cdot\mathbf{i} = 0,\ \mathbf{R}\cdot\mathbf{j} = 0$$

衝撃の方向

$$\left(\mathbf{v}_2' + \boldsymbol{\omega}_2' \times \mathbf{p}_2 - \mathbf{v}_1' - \boldsymbol{\omega}_1' \times \mathbf{p}_2\right)\cdot\mathbf{k} = 0$$

衝突点における速度の差がゼロ



衝突の計算（4）

- 衝突点での衝撃 R と、衝突後の両物体の速度・回転速度の方程式を解く
  - 式全体を15×15次元の行列として表し、逆行列を解くことで、全ての未知数が計算できる
    - 逆行列は、LU分解などの一般的な方法で計算可能

$$\begin{pmatrix} m_1 & 0 & 0 & 0 & -1 \\ 0 & m_2 & 0 & 0 & 1 \\ 0 & 0 & \mathbf{I}_1 & 0 & -\mathbf{p}_1^* \\ 0 & 0 & 0 & \mathbf{I}_2 & \mathbf{p}_2^* \\ -1 & 1 & -\mathbf{p}_1^* & \mathbf{p}_2^* & \mathbf{i},\mathbf{j},\mathbf{k} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1' \\ \mathbf{v}_2' \\ \boldsymbol{\omega}_1' \\ \boldsymbol{\omega}_2' \\ \mathbf{R} \end{pmatrix} = \begin{pmatrix} m_1\mathbf{v}_1 \\ m_2\mathbf{v}_2 \\ \mathbf{I}_1\boldsymbol{\omega}_1 \\ \mathbf{I}_2\boldsymbol{\omega}_2 \\ 0 \end{pmatrix}$$

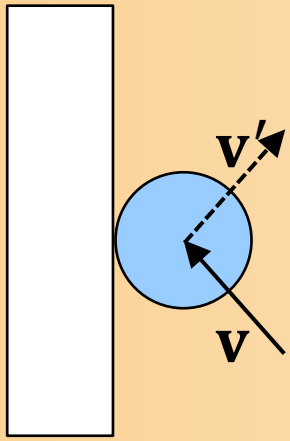
$$\mathbf{p}^* = \begin{pmatrix} 0 & -\mathbf{p}_z & -\mathbf{p}_y \\ -\mathbf{p}_z & 0 & -\mathbf{p}_x \\ -\mathbf{p}_y & -\mathbf{p}_x & 0 \end{pmatrix}$$

一番下の行は適当（正しく書くと細かくなるため）

外積計算を表す行列

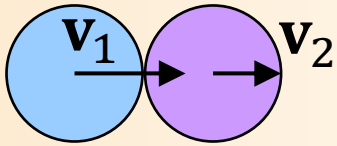
参考：簡易的な衝突計算

- 物体の回転を考慮せず、並進速度のみの計算であれば、簡単な式で計算できる
  - 高校物理レベルの計算



$$\mathbf{v}'_x = -\mathbf{v}_x$$

$$\mathbf{v}'_y = \mathbf{v}_y$$



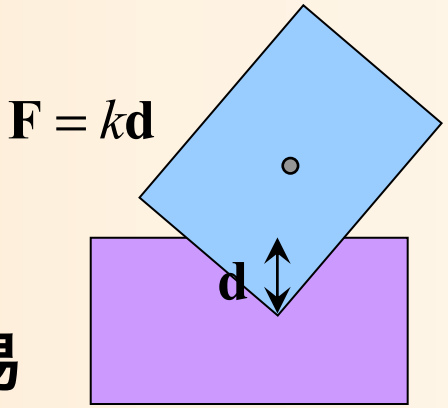
$$m_1\mathbf{v}_1 + m_2\mathbf{v}_2 = m_1\mathbf{v}'_1 + m_2\mathbf{v}'_2$$

$$\frac{\mathbf{v}'_1 - \mathbf{v}'_2}{\mathbf{v}_1 - \mathbf{v}_2} = 1$$

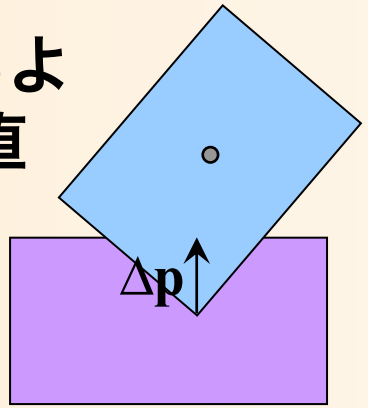
こういった簡易的な衝突計算は、限られた用途にしか使えず、一般的には正確な衝突計算を行う必要がある

接触の計算

- 主に２種類の方法がある
- ペナルティ法
  - めり込みの深さに応じて力を加える
  - 適当なバネ係数を決める
  - 正確さは保障されないが、処理は容易



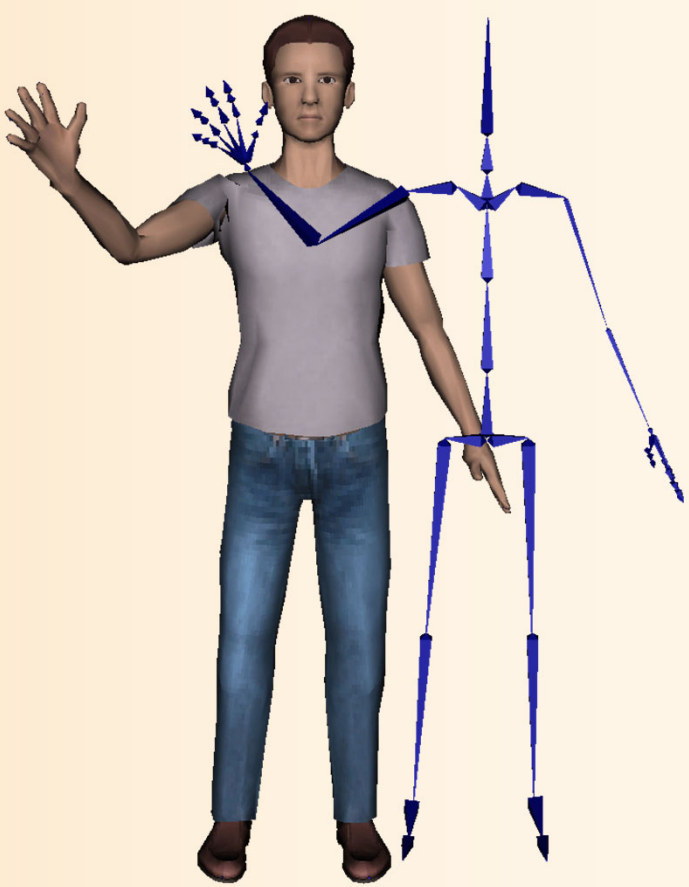
- コンストレイント法（制約法）
  - めり込みが起こらない位置まで移動するような力を加える （または位置・速度を直接修正）
  - 計算が複雑になる、物理法則が崩れる
  - 複雑なめり込みに対処するのが困難



多関節体の物理シミュレーション

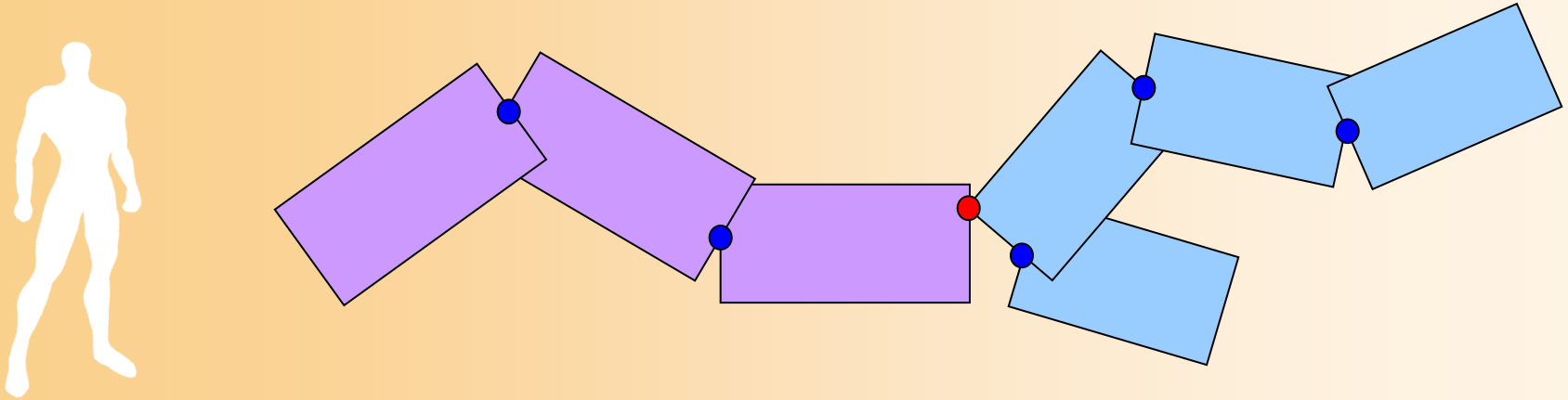
多関節体のシミュレーション

- 運動方程式
  - 基本的には、複数の剛体として扱うことができる
  - 関節点での拘束条件（関節で剛体同士が常に接触）を考慮
  - 各種の運動を計算するときは、関節の拘束条件を考慮し、全身で運動を計算する必要がある

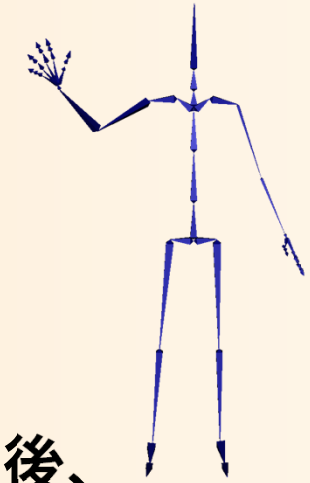


多関節体の衝突計算

- 高次元の行列計算
  - 衝突点における条件の式に加えて、全関節における拘束条件の式を追加し、連立方程式として解く必要がある
    - 関節点での未知数15 ＋ 互いの関節数×6
  - 高次元の疎行列になるので、疎行列に向いたデータ構造や計算方法が使える

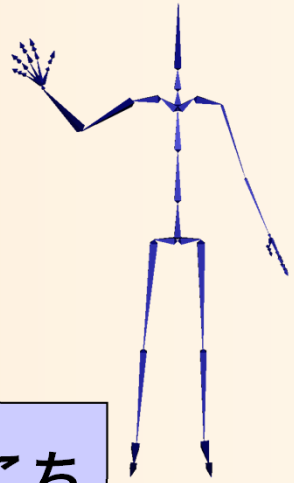


多関節体のシミュレーション方法



- 主に２種類の方法がある
- 剛体シミュレーションにより計算
  - 各リンクの位置・向きにより状態を表現
  - 各リンクの運動をばらばらに計算し、その後、関節間の拘束条件を保つように制約を適用する
- 多関節体全体で計算
  - 各関節の関節角度により状態を表現
  - 常に拘束条件を満たすように全身の動きを計算
  - 結果的に、計算に時間がかかってしまう

多関節体のシミュレーション方法



- 主に２種類の方法がある
- 剛体シミュレーションにより計算
  - 各リンクの位置・向きにより状態を表現

現在は、単純な物理シミュレーションだけの用途であれば、こちらの方法が主流

- 多関節体全体で計算
  - 各関節の関節角度により状態を表現
  - 拘束条件を満たすよう、全身の動きを計算

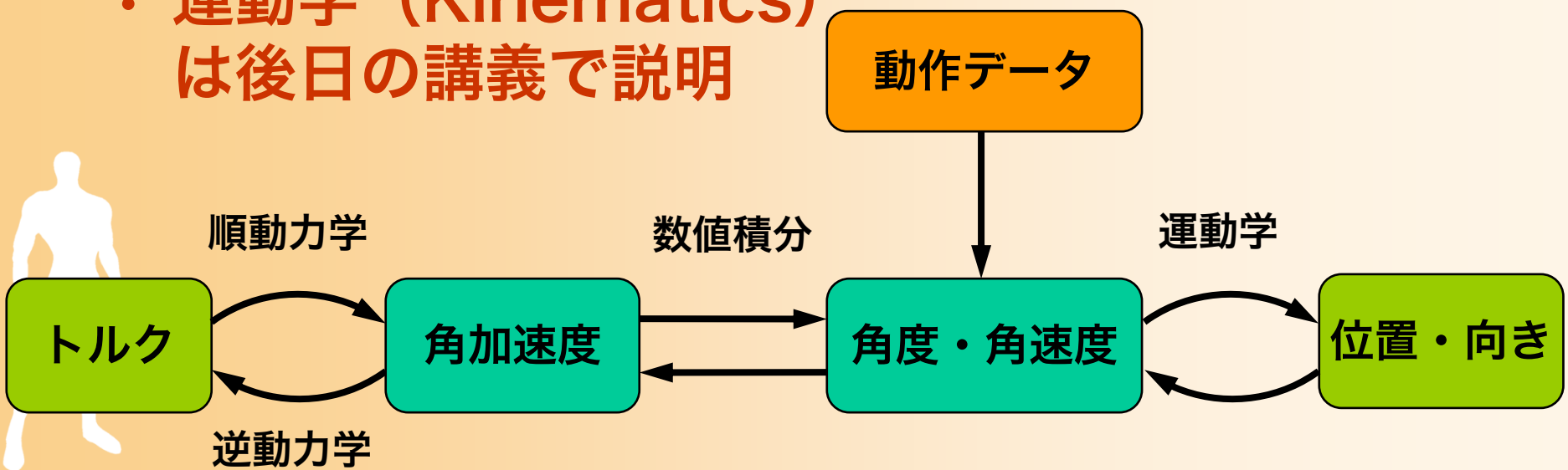
以降の説明は、主にこちらの方法での説明
ロボット制御などで、多関節体の運動を解析するためには、こちらの方法も必須

多関節体の運動計算

- 順動力学（Forward Dynamics）
  - 全関節に加わるトルクから、運動を計算する
  - 計算には比較的時間がかかる
- 逆動力学（Inverse Dynamics）
  - 目標として与えられた運動（角加速度）から、その運動の実現に必要なトルクを逆算する
  - ロボットや人体モデルの制御などで必要になる
  - こちらは比較的高速に計算できる

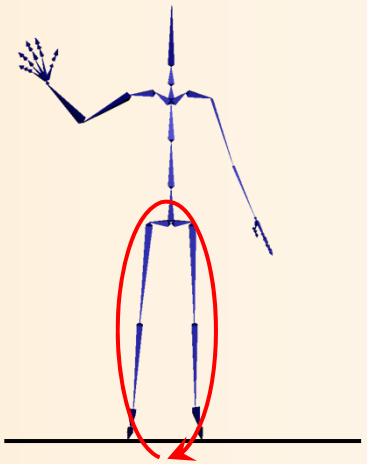
多関節体の動力学

- 順動力学（Forward Dynamics）
  - 全関節のトルクから加速度の変化を計算
- 逆動力学（Inverse Dynamics）
  - 全関節の加速度から必要なトルクを計算
- 運動学（Kinematics）は後日の講義で説明



多関節体の逆動力学計算

- 開ループ構造の逆動力学計算
- 主に2つの解法がある
  - ラグランジュ法
    - 各体節の運動エネルギーをもとに計算
  - ニュートン・オイラー法
    - 各体節の加速度とトルクにより計算
    - 高速



閉ループ

- 閉ループ構造の場合は、拘束条件を考慮して次元を減らした状態で計算する必要がある

多関節体の逆動力学計算

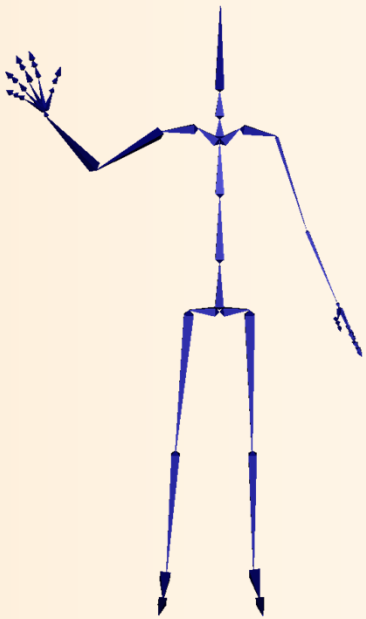
- 逆動力学計算の式

$$\tau = H(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) + K(\theta)F$$

必要トルク　　角加速度

- ニュートン・オイラー法

1. 順方向計算
   - 支点から末端に向かって、各関節の加速度を加算
2. 逆方向計算
   - 末端から支点に向かって、各関節の必要トルクを減算



多関節体の順動力学計算

- 単純な計算方法
  - 逆動力学計算を繰り返すことによって、逆動力学計算式の各行列を求めることができる

$$\tau = H(\theta)\ddot{\theta} + C(\theta,\dot{\theta}) + G(\theta) + K(\theta)F$$

  - 逆行列を計算することで、順動力学の式を得る

$$\ddot{\theta} = H(\theta)^{-1}\left(C(\theta,\dot{\theta}) + G(\theta) + K(\theta)F - \tau\right)$$

- より高速な計算方法もある（説明は省略）

多関節体の動作生成

- 物理シミュレーション自体は、順動力学により実現可能
- 人間の場合、どのような状況でどのようなトルクが関節に生じるかという運動モデルが未知

$$\ddot{\theta} = H(\theta)^{-1}\left(C\left(\theta,\dot{\theta}\right) + G(\theta) + K(\theta)F - \tau\right)$$

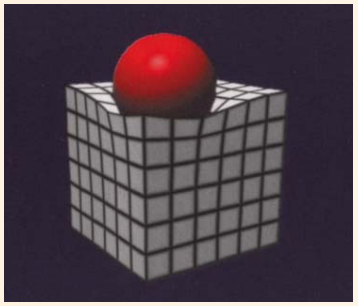
  - 高いところから落ちる動作など、ほとんど自力で運動できない状況であれば、シミュレーション可能
  - 比較的単純な動作であれば、ロボットのコントローラなどを応用することで、実現可能

- 人間らしい運動の実現には、運動モデルや筋肉モデル（トルク特性）を考慮する必要がある

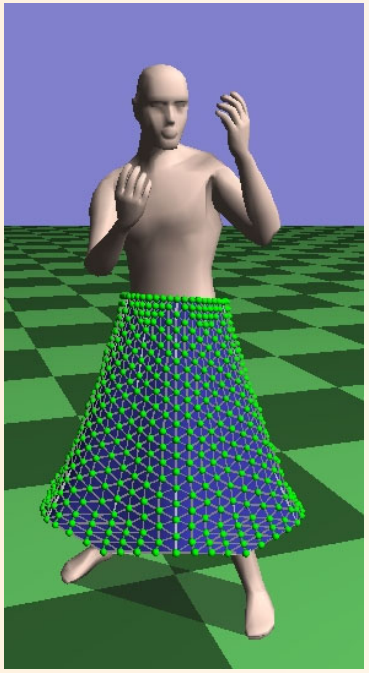
変形する物体の
物理シミュレーション

変形する物体の運動計算

- 変形する物体
  (Deformable Object)
  - 衣服、皮膚、髪の毛など
  - 粒子モデルや有限要素法などの手法
    - 物体を細かい要素に分ける
    - 各時刻において、各要素に働く力を計算
      - 有限要素法では、要素に働く力を積分により求める
      - 粒子モデルでは、要素を点とみなして力を計算
    - 力から変形（加速度）を計算
- 詳しい説明は省略

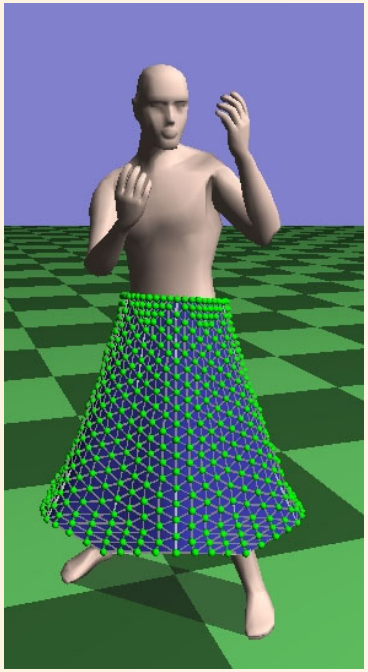


[Terzopoulos 87]



変形する物体の運動計算の例

- 粒子モデルによる衣服シミュレーション
  - 衣服を格子状の粒子（質点）のつながりで表す
    - 隣り合う粒子間は、ばねでつながっているものとする
  - 粒子に働く力を定義
    - 隣接する粒子間の長さを一定に保とうとする力（ばねの力）
    - 衣服が曲がったときに戻ろうとする力
    - 重力
    - 人体からの反発力
  - 各フレームにおける力の和から、粒子の加速度を計算



物理シミュレーションの最新技術

- 物理シミュレーションの原理は古くから確立
- 複雑な状況でも高速・安定した計算を実現することは、現在でも難しい課題
  - 特に衝突・接触が多く発生するような状況
- データにもとづくシミュレーション
  - 与えられたデータにもとづいて高速なシミュレーションを実現
    - 次元削減や既存のデータをつなげた結果生成など
  - 機械学習を応用した人体の運動モデル

まとめ

- 物理シミュレーションの概要
- 剛体の物理シミュレーション
  - 運動方程式
  - 回転運動と慣性モーメント
  - シミュレーションの手順
- 衝突と接触の扱い
- 多関節体の物理シミュレーション
- 変形する物体の物理シミュレーション

次回予告

- 衝突判定
  - 近似形状による衝突判定
  - 空間インデックス
  - ポリゴンモデル同士の衝突判定
- ピッキング
  - サンプルプログラム
  - スクリーン座標系での判定
  - ワールド座標系での判定
  - レポート課題