物理エンジンの作り方

株式会社コーエー
技術支援部シニアエキスパート
津田順平

このセッションで取り扱う範囲

- 剛体の物理
- 拘束なしの運動
- 拘束ありの運動
- 安定化/高速化の技法

コリジョンは扱いません

- コリジョンについては以下を受講下さい

- ＣＥＤＥＣラボ
  「衝突判定法をソート･検索アルゴリズムの観点から眺める」　講師：長谷川先生（電通大）

- 日時：10/10(水) 16:40

なぜ自社開発か？

- 上位レイヤーを作ることが目的
  - より高度なキャラクタ制御など
  - 基盤技術はブラックボックスにしたくない

- テクノロジーの深い理解
  - 自らが使う技術を「作れる」ことは重要
  - 外部のミドルウェアを使う場合にも有利

剛体の物理 (拘束なしの場合)

並進の運動方程式

並進運動の運動方程式

$\mathbf{F} = m\dot{\mathbf{v}}$　　　　(**F**:力, m:質量, $\dot{\mathbf{v}}$:加速度)

実際にプログラムで使う場合は

$\dot{\mathbf{v}} = \mathbf{F}/m$

並進運動の解法

- 数値積分で時間発展を追いかけていく

- 速度の更新

$$\mathbf{v}_{t+1} = \Delta t\, \dot{\mathbf{v}}_t + \mathbf{v}_t$$

- 位置の更新

$$\mathbf{p}_{t+1} = \Delta t\, \mathbf{v}_t + \mathbf{p}_t$$

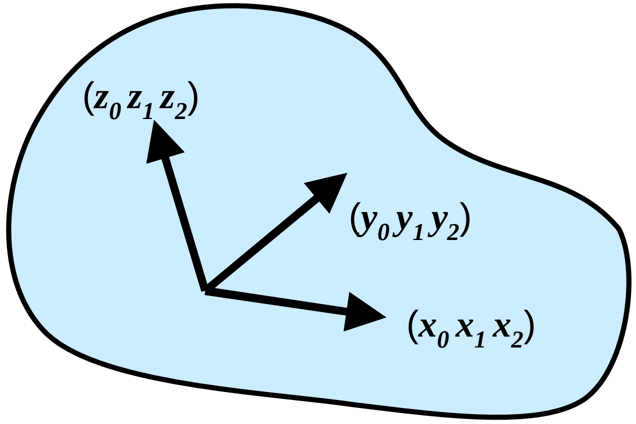
剛体

- 質点
  - 必要な情報は「位置」のみ
- 剛体
  - 新たに「向き」の情報が必要

剛体の向き

姿勢マトリクスで表現

$$\mathbf{R} = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{bmatrix}$$



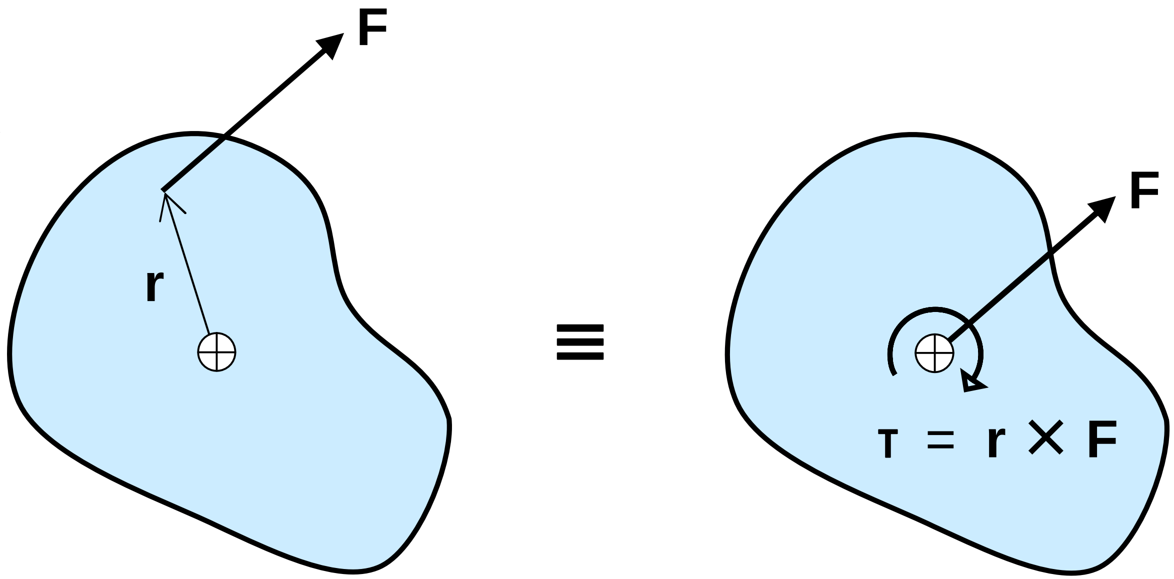
剛体に固定されたローカル座標系に対応
列ベクトルは単位ベクトル
列ベクトルは互いに直交

トルク(ひねり力)

定義: $\tau = \mathbf{r} \times \mathbf{F}$

(r：重心から作用点に向かうベクトル, F:力)

τの向き＝ひねる軸
τの大きさ＝ひねる強さ



慣性テンソル

ひねり力に対する「抵抗」

3ｘ3の行列で表現

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

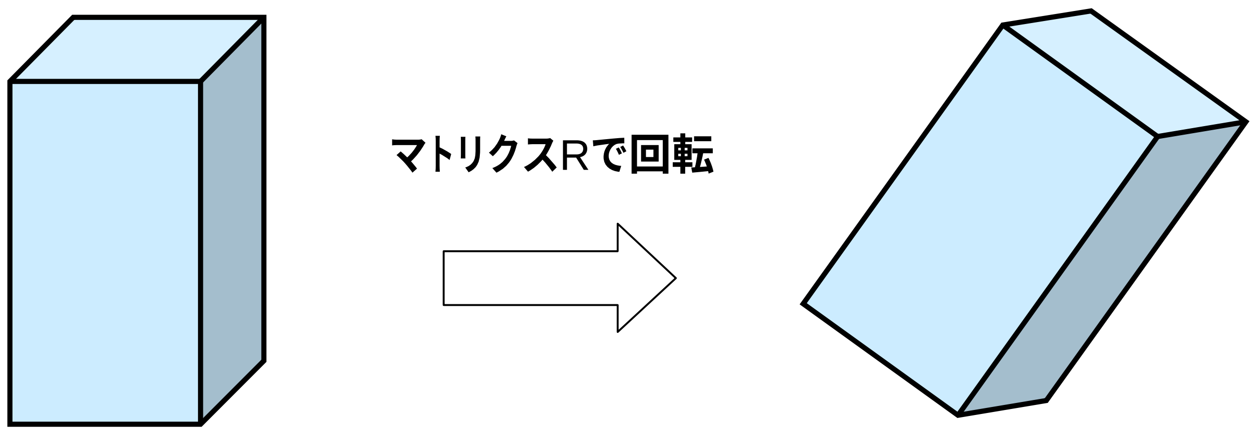
R :姿勢マトリクス

$\mathbf{R}^{T}$:Rの転置マトリクス

I′ :基準姿勢での慣性テンソル(基本形状ごとに公式あり)

慣性テンソル

質量 M, 辺長 x,y,z の直方体の場合



$$\mathbf{I}' = 1/12M \begin{pmatrix} y^2+z^2 & 0 & 0 \\ 0 & z^2+x^2 & 0 \\ 0 & 0 & x^2+y^2 \end{pmatrix}$$

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

並進運動と回転運動

並進	回転
F (力)	τ (トルク)
m(質量)	I (慣性テンソル)
p(位置)	R (向き)
v(速度)	ω (角速度)
$\dot{v}$(加速度)	$\dot{\omega}$ (角加速度)

並進運動と回転運動

並進	回転
$\mathbf{P} = m\mathbf{v}$ (運動量の定義)	$\mathbf{L} = \mathbf{I}\boldsymbol{\omega}$ (角運動量の定義)
$\mathbf{F} = \dot{\mathbf{P}}$ (運動量の変化式)	$\boldsymbol{\tau} = \dot{\mathbf{L}}$ (角運動量の変化式)
$\mathbf{F} = m\dot{\mathbf{v}}$ (Newtonの運動方程式)	$\boldsymbol{\tau} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}$ (Eulerの運動方程式)

剛体回転の複雑さ

Eulerの運動方程式がヒント

重力しか作用していない場合 $\tau = 0$

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

代入して変形すると

$$\dot{\boldsymbol{\omega}} = -\mathbf{I}^{-1}(\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega})$$

トルクが働かない場合でも角加速度 $\dot{\boldsymbol{\omega}} \neq 0$

つまり回転の軸、大きさが変化する！

回転運動に関する方程式

- Eulerの運動方程式
  - 高精度の数値積分のテクニックが必要
  - 単純に解くと角速度が容易に発散
- 角運動量の変化式
  - 角運動量保存則を反映
  - 単純に解いても安定した結果

回転運動の解き方

Step1 <姿勢の更新>

$$\mathbf{R}_{t+1} = \Delta t\, \dot{\mathbf{R}}_t + \mathbf{R}_t \qquad (\dot{\mathbf{R}}_t = \boldsymbol{\omega}_t \times \mathbf{R}_t)$$

Step2 <慣性テンソルの更新>

$$\mathbf{I}_{t+1} = \mathbf{R}_{t+1}\, \mathbf{I}'\, \mathbf{R}_{t+1}{}^{\mathrm{T}}$$

Step3 <角運動量の更新>

$$\mathbf{L}_{t+1} = \mathbf{L}_t + \Delta t\, \boldsymbol{\tau}_t \qquad (\dot{\mathbf{L}} = \boldsymbol{\tau})$$

Step4 <角速度の更新>

$$\boldsymbol{\omega}_{t+1} = \mathbf{I}^{-1}{}_{t+1} \mathbf{L}_{t+1} \qquad (\mathbf{L} = \mathbf{I}\boldsymbol{\omega})$$

なぜ安定する？

- 自由落下の際は Step3 で τ = 0
- 角運動量 L が変わらないことが保証される
- Euler の運動方程式は単純に解くと L が増大

  ∵式中で角運動量は明示的には扱われないので

物理シミュレーションのレベル

- 拘束なしの運動
  - これまでの話ですべてです
- 拘束ありの運動
  - 物理エンジンの実装上もっとも難しい部分
  - コードの9割以上を占める (弊社の場合)

剛体の物理 (拘束ありの場合)

拘束ありの運動

- 貫通の防止
  - テーブル面より下にコーヒーカップは沈まない
- ジョイントの処理
  - ヒンジ (ひじ/ひざの回転、ドアの開閉)
  - ボールジョイント (首/腰の回転)
- 可動範囲の制限
  - 首は真後ろには向かない

拘束の解き方

- インパルス法
  - 拘束を多数の小さな衝突 (インパルス) として扱う
- 解析法
  - 拘束を表す連立方程式を解いて拘束力を求める
  - 本セッションではこちらを取り扱います

拘束の解き方

名称については混乱気味

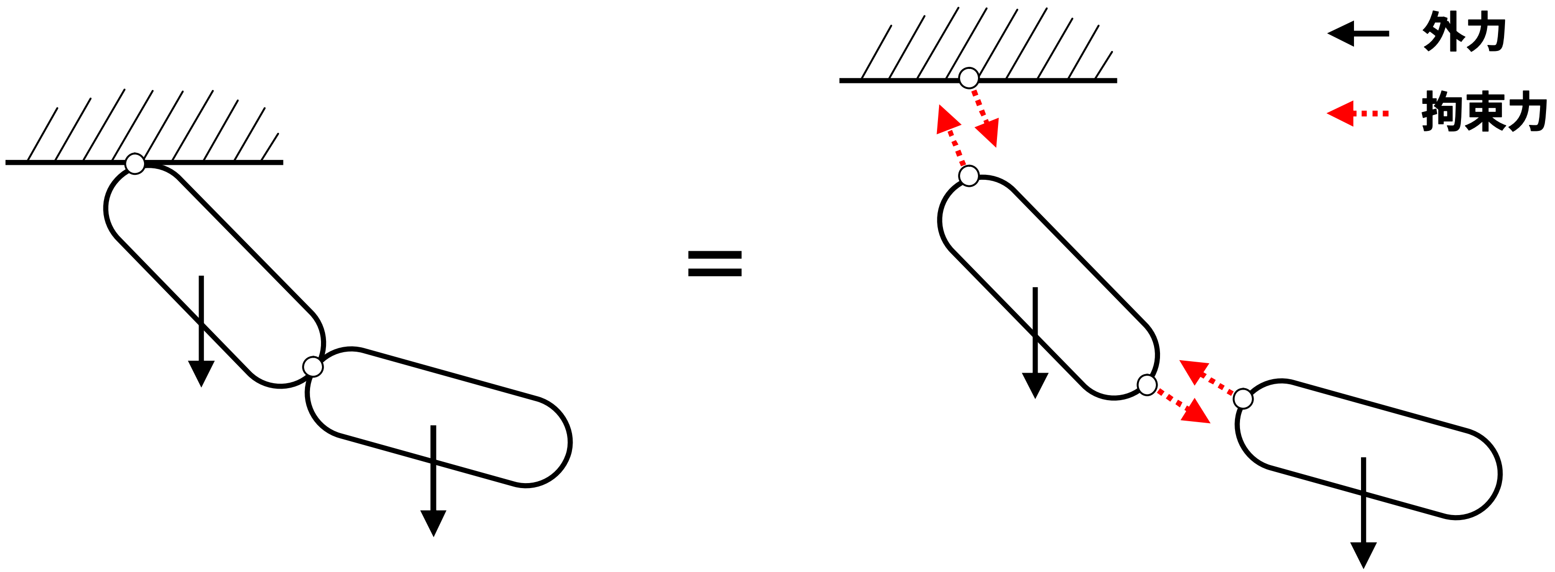
Erleben's Paper [Erleben07]	Garstenauer's Thesis [Garst03]	ＣＥＤＥＣなど
インパルスベース法 (Impulse Based Method)	逐次インパルス法 (Sequential Impulse Based Method)	インパルス法
拘束ベース法 (Constraint Based Method)	同時インパルス法 (Simultanius Impuse Based Method)	解析法

拘束をどう取り扱うか？（解析法）

拘束を実現する力、「拘束力」を求める

拘束力を運動方程式の F、τ に算入

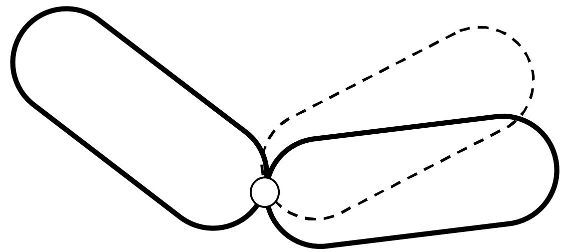
後は拘束なしの問題として解けばよい



拘束をどう定式化するか？

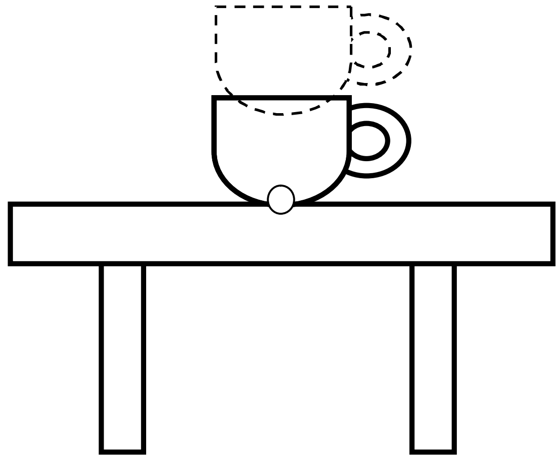
等式拘束

上腕の下端部 = 下腕の上端部



不等式拘束

カップの下面 ≧ テーブルの上面



拘束をどんな量で表現するか？

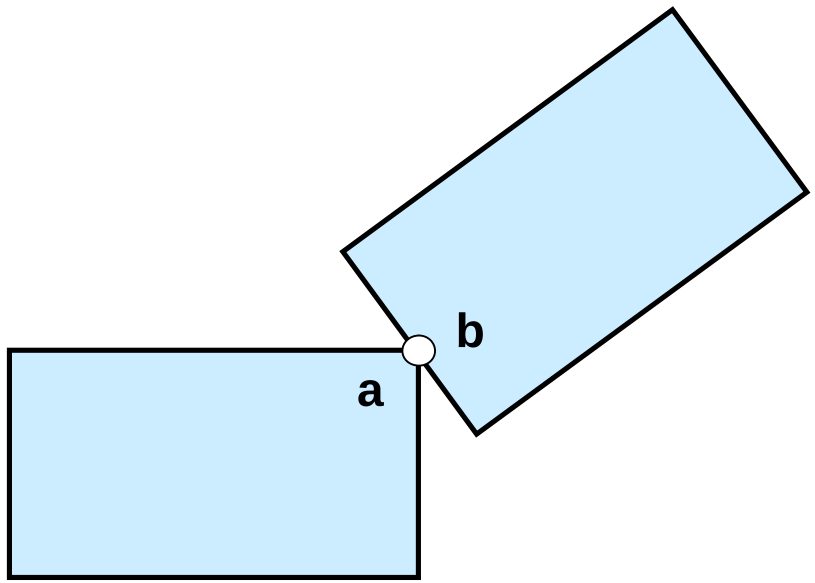
- 位置
  - 結果の補正処理に利用 [Elreben05]
- 速度
  - 現在では (おそらく) これが定番 [Elreben05]
- 加速度
  - 昔はこれで表現していました [Baraff 89]

速度による拘束の表現 (ジョイント)

2物体の a 点と b 点が接合されている.

2点の速度をそれぞれ $\mathbf{v}_a$, $\mathbf{v}_b$ とすると

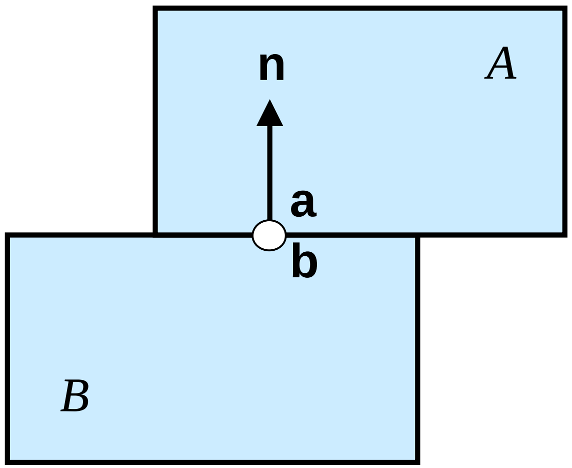
$$\mathbf{v}_a = \mathbf{v}_b$$



速度による拘束の表現 (貫通なし)

物体 A が物体 B の上に載っている.
接触点を **a**, **b**, 接触点での B 側法線を **n** ,
2点の速度をそれぞれ $\mathbf{v}_a$, $\mathbf{v}_b$ とすると

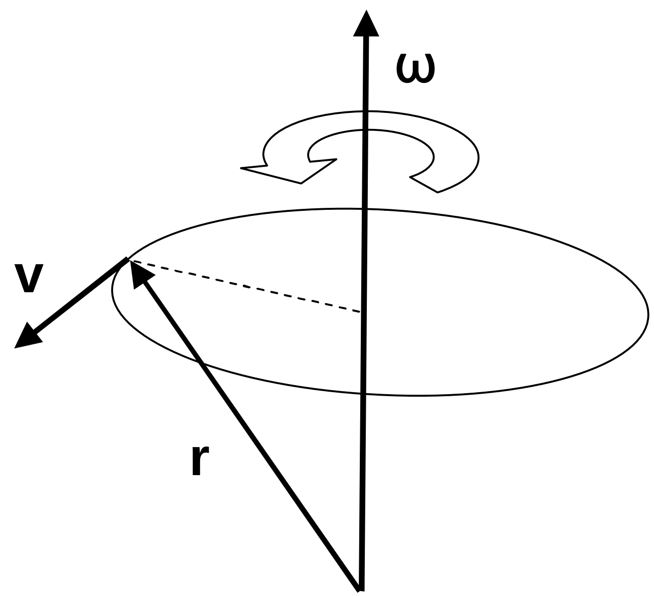
$$\mathbf{n} \cdot (\mathbf{v}_a - \mathbf{v}_b) \geqq 0$$



角速度→並進速度

ベクトル r が角速度 ω で回転しているとき
r の先端点の接線方向の並進速度 v は

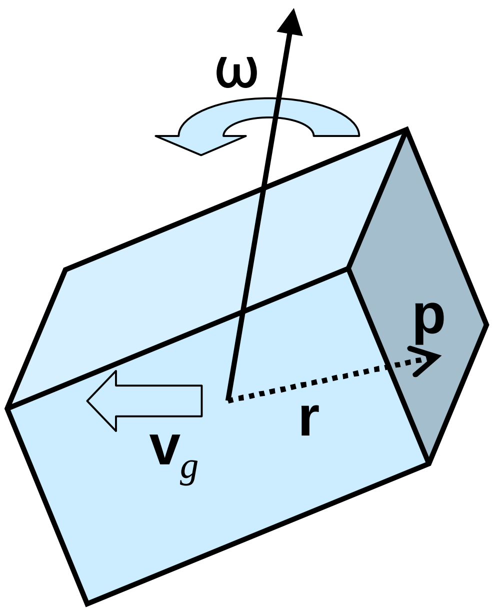
$$\mathbf{v} = \boldsymbol{\omega} \times \mathbf{r}$$



剛体上の点の速度

剛体の角速度ベクトルを ω, 重心速度を $\mathbf{v}_g$
重心から剛体上の点 p に向かうベクトルを r とすると
点 p の速度 $\mathbf{v}_p$ は

$$\mathbf{v}_p = \mathbf{v}_g + \boldsymbol{\omega} \times \mathbf{r}$$



一般化速度

- 剛体上の点の速度を表現するには速度v と角速度ωが必要
- 剛体の場合 v, ω のセットで「完全」な速度情報となる
- これより”一般化速度” u を以下のように定義する

$$\mathbf{u} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

質量マトリクス

質量と慣性テンソルもセットにして表現

$$
\mathrm{M} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & I_{xy} & I_{xz} \\ 0 & 0 & 0 & I_{yx} & I_{yy} & I_{yz} \\ 0 & 0 & 0 & I_{zx} & I_{zy} & I_{zz} \end{bmatrix}
$$

一般化速度と質量マトリクス

質量マトリクスと一般化速度をかけると

$$\mathbf{Mu} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & I_{xy} & I_{xz} \\ 0 & 0 & 0 & I_{yx} & I_{yy} & I_{yz} \\ 0 & 0 & 0 & I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{pmatrix} m\mathbf{v} \\ \mathbf{I\omega} \end{pmatrix} = \begin{pmatrix} \mathbf{P} \\ \mathbf{L} \end{pmatrix}$$

運動量と角運動量がセットで得られる

ヤコビアン

とりあえずここでは、
”異なる座標系間での速度の「変換」を行う演算子”
ぐらいに考えてください

$$\mathbf{v}^A = \mathbf{J}\ \mathbf{v}^B$$

$\mathbf{v}^A$ ：ある運動を座標系Aで記述した速度

$\mathbf{v}^B$ ：運動を座標系Bで記述した速度

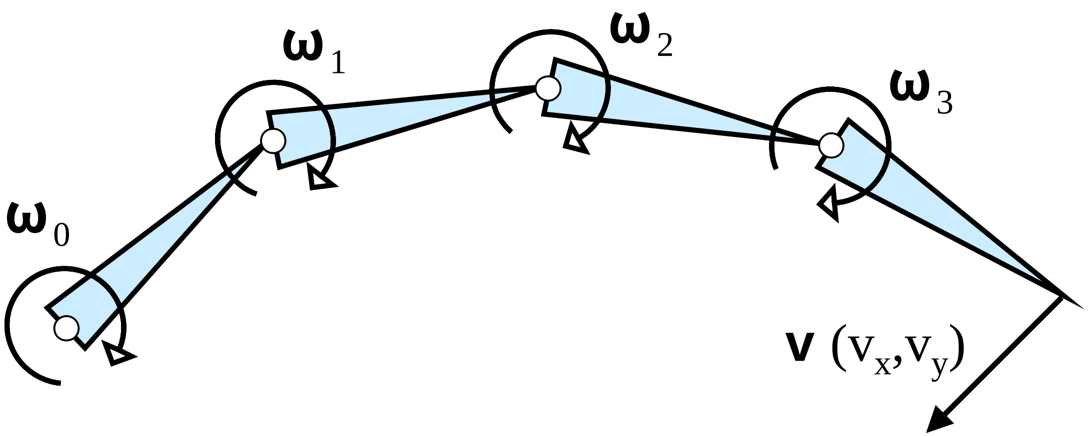
J ：ヤコビアン

ヤコビアン

例 : 関節の角速度 → 手先の並進速度

(4次元)　　　　　　　　　　(2次元)

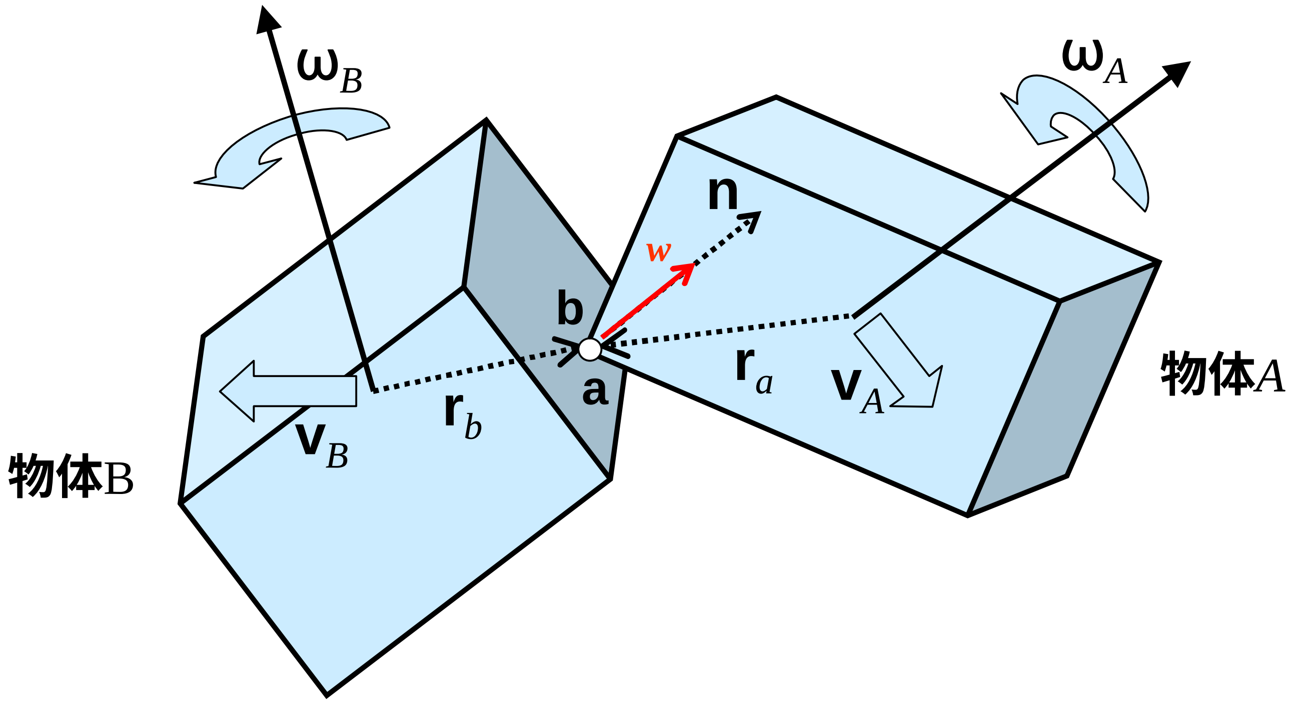
$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \mathbf{J} \begin{bmatrix} \boldsymbol{\omega}_0 \\ \boldsymbol{\omega}_1 \\ \boldsymbol{\omega}_2 \\ \boldsymbol{\omega}_3 \end{bmatrix}$$



拘束軸上の速度

物体*A*, *B* がそれぞれの所属点 a, b で接触している

*B*側の面法線 n 上での接触点 a, b の相対速度 *w* は？

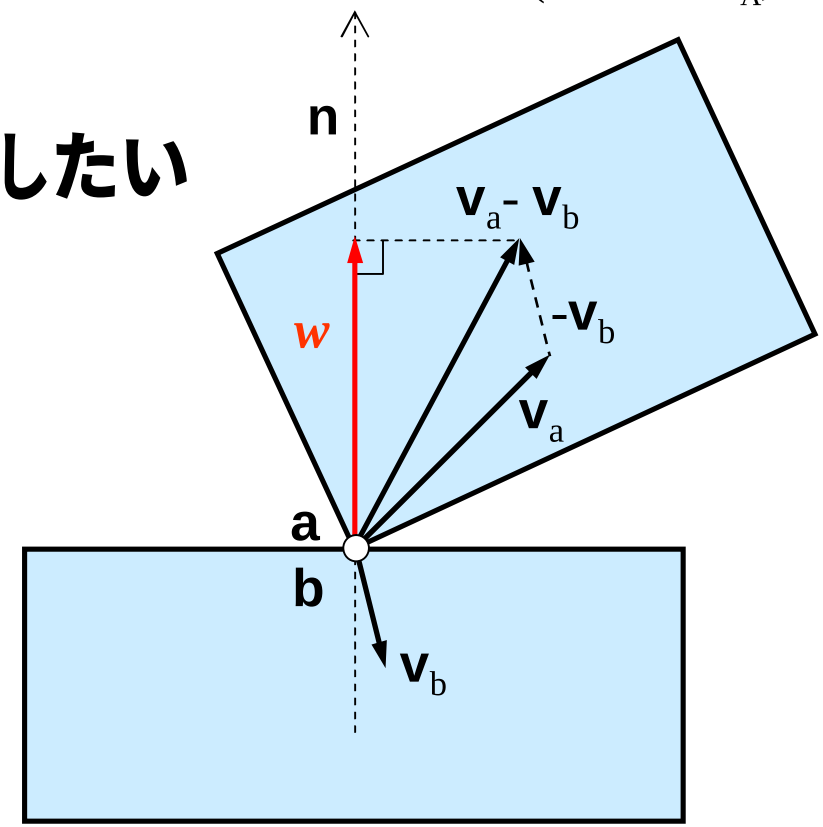


拘束軸上の速度

相対速度 w を

$\mathbf{v}_A, \omega_A, \mathbf{v}_B, \omega_B$ を使って表現したい

物体A(重心速度$\mathbf{v}_A$,角速度$\boldsymbol{\omega}_A$)



物体B(重心速度$\mathbf{v}_B$,角速度$\boldsymbol{\omega}_B$)

拘束軸上の速度

接触点 a, b の速度は

$$\mathbf{v}_a = \mathbf{v}_A + \boldsymbol{\omega}_A \times \mathbf{r}_a\,,\quad \mathbf{v}_b = \mathbf{v}_B + \boldsymbol{\omega}_B \times \mathbf{r}_b$$

点 b から見た点 a の相対速度は

$$\mathbf{v}_{rel} = \mathbf{v}_a - \mathbf{v}_b$$

拘束軸 n に投影した速度 w は

$$w = \mathbf{n} \cdot \mathbf{v}_{rel}$$

以上をまとめると

$$w = \mathbf{n} \cdot (\mathbf{v}_A + \boldsymbol{\omega}_A \times \mathbf{r}_a - \mathbf{v}_B - \boldsymbol{\omega}_B \times \mathbf{r}_b)$$

拘束軸上の速度

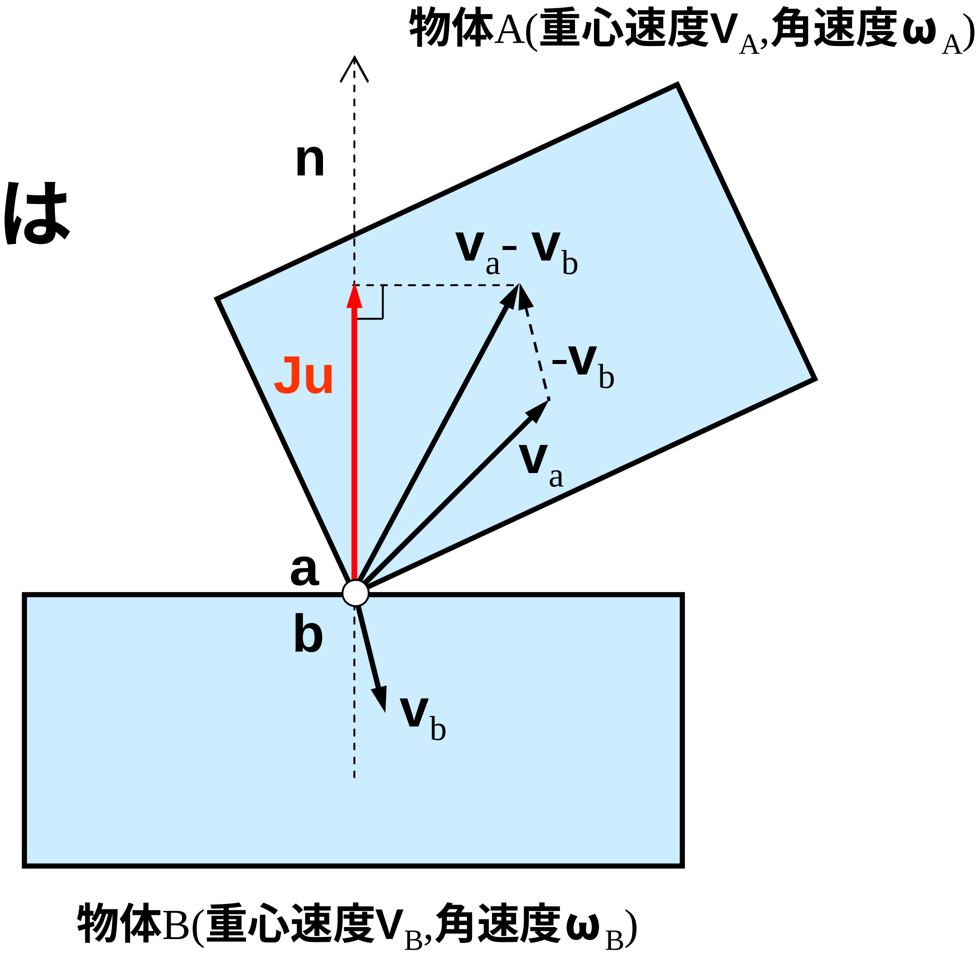
$$
\begin{aligned}
w &= \mathbf{n}\cdot(\mathbf{v}_A + \boldsymbol{\omega}_A\times\mathbf{r}_a - \mathbf{v}_B - \boldsymbol{\omega}_B\times\mathbf{r}_b) \\
&= \mathbf{n}\cdot\mathbf{v}_A + \mathbf{n}\cdot\boldsymbol{\omega}_A\times\mathbf{r}_a - \mathbf{n}\cdot\mathbf{v}_B - \mathbf{n}\cdot\boldsymbol{\omega}_B\times\mathbf{r}_b \\
&= \mathbf{n}\cdot\mathbf{v}_A + \mathbf{r}_a\times\mathbf{n}\cdot\boldsymbol{\omega}_A - \mathbf{n}\cdot\mathbf{v}_B - \mathbf{r}_b\times\mathbf{n}\cdot\boldsymbol{\omega}_B \\
&= (\mathbf{n}^\mathrm{T}, (\mathbf{r}_a\times\mathbf{n})^\mathrm{T}, -\mathbf{n}^\mathrm{T}, -(\mathbf{r}_b\times\mathbf{n})^\mathrm{T}) \begin{pmatrix} \mathbf{v}_A \\ \boldsymbol{\omega}_A \\ \mathbf{v}_B \\ \boldsymbol{\omega}_B \end{pmatrix} \\
&= (\mathbf{J}_A, \mathbf{J}_B) \begin{pmatrix} \mathbf{u}_A \\ \mathbf{u}_B \end{pmatrix} \\
&= \mathbf{J}\mathbf{u}
\end{aligned}
$$

拘束の表現

したがって、
貫通しないという拘束条件は

$$Ju \geqq 0$$

という式で表現できる



拘束の表現

一般の場合でも拘束条件は

$Ju \geqq 0$　　(貫通なし、可動範囲制限など)

または

$Ju = 0$　　(ジョイントなど)

として表現できる

拘束力の算定法

$$\mathbf{Ju} = (\mathbf{n}^{\mathrm{T}},\ (\mathbf{r}_a \times \mathbf{n})^{\mathrm{T}},\ -\mathbf{n}^{\mathrm{T}},\ -(\mathbf{r}_b \times \mathbf{n})^{\mathrm{T}}) \begin{bmatrix} \mathbf{v}_A \\ \boldsymbol{\omega}_A \\ \mathbf{v}_B \\ \boldsymbol{\omega}_B \end{bmatrix} \geqq 0 \quad \text{(貫通なし拘束)}$$

- 知りたいのは「拘束力」
- 今のところ拘束条件式には拘束力に相当する項はない
- もしあれば拘束力を未知数として条件式を解けばよい
- どのように拘束力の項を導入するか？

拘束力の方向と大きさ

- 力は「方向」と「大きさ」で表現できる
- 実は拘束力の「方向」はわかっている
- 「大きさ」だけが未知数

拘束力の方向

拘束条件を示すJが求まったとすると

拘束力の方向 = J^T

なぜか？

・Jはその方向には「動けない」軸を示している

・動ける方向には「抵抗」がないので力は発生しない

・動けない方向にのみ力は発生

数理的な詳細は[Catt08]

条件式への拘束力の導入

1タイムステップ先の速度は以下のとおり

$$\mathbf{u}' = \mathbf{u} + \dot{\mathbf{u}} \times \Delta t$$
$$= \mathbf{u} + \mathbf{M}^{-1}(\mathbf{F}_{external} + \mathbf{F}_{constraint})\Delta t$$

1タイムステップ先でも拘束は成立（消失しない限り）

$$\mathbf{J}\mathbf{u}' = \mathbf{J}\mathbf{u} + \mathbf{J}\mathbf{M}^{-1}(\mathbf{F}_{external} + \mathbf{F}_{constraint})\Delta t \geqq 0$$

拘束力の方向は$\mathbf{J}^T$,その大きさをλとすると

$$\mathbf{J}\mathbf{u}' = \mathbf{J}\mathbf{u} + \mathbf{J}\mathbf{M}^{-1}(\mathbf{F}_{external} + \mathbf{J}^T\lambda)\Delta t \geqq 0$$
$$= \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda\Delta t + \mathbf{J}(\mathbf{u} + \mathbf{M}^{-1}\mathbf{F}_{external}\Delta t) \geqq 0$$

拘束条件式の一般形

簡単のために Δt を $\boldsymbol{\lambda}$ の中に含めると

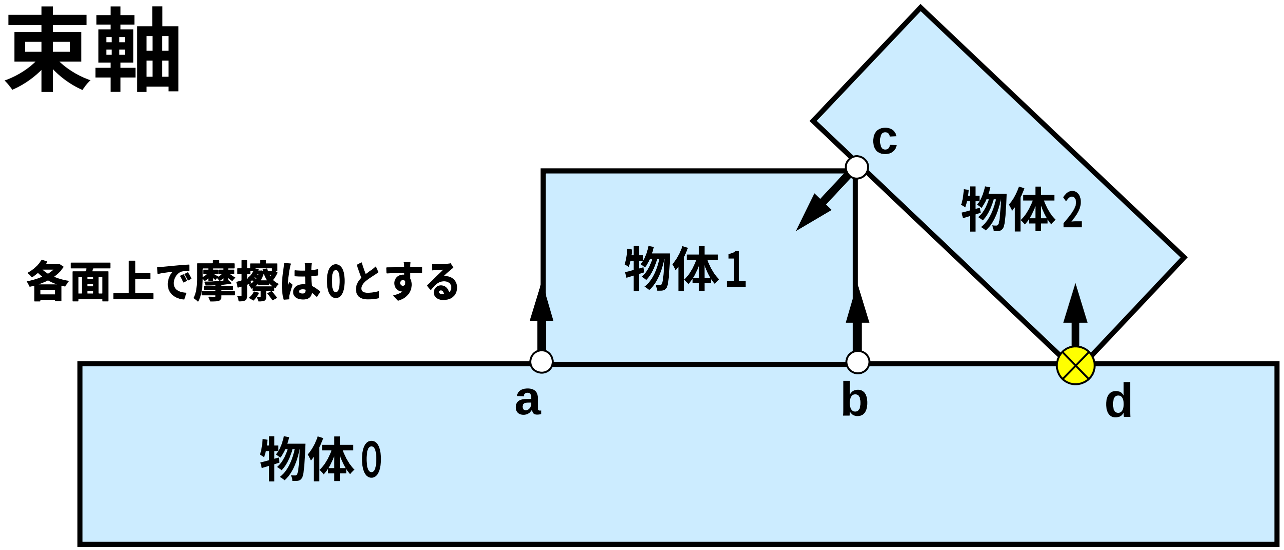
$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda} + \mathbf{b} \geqq 0$ （貫通なし、可動制限など）

または

$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda} + \mathbf{b} = 0$ （ジョイントなど）

多体の物理

- 面上拘束 ＝ 等式拘束
- 接触点 ＝ 不等式拘束
- 拘束軸



多体の物理 (拘束マトリクス)

系全体の拘束マトリクスを組み立てる

物体の数

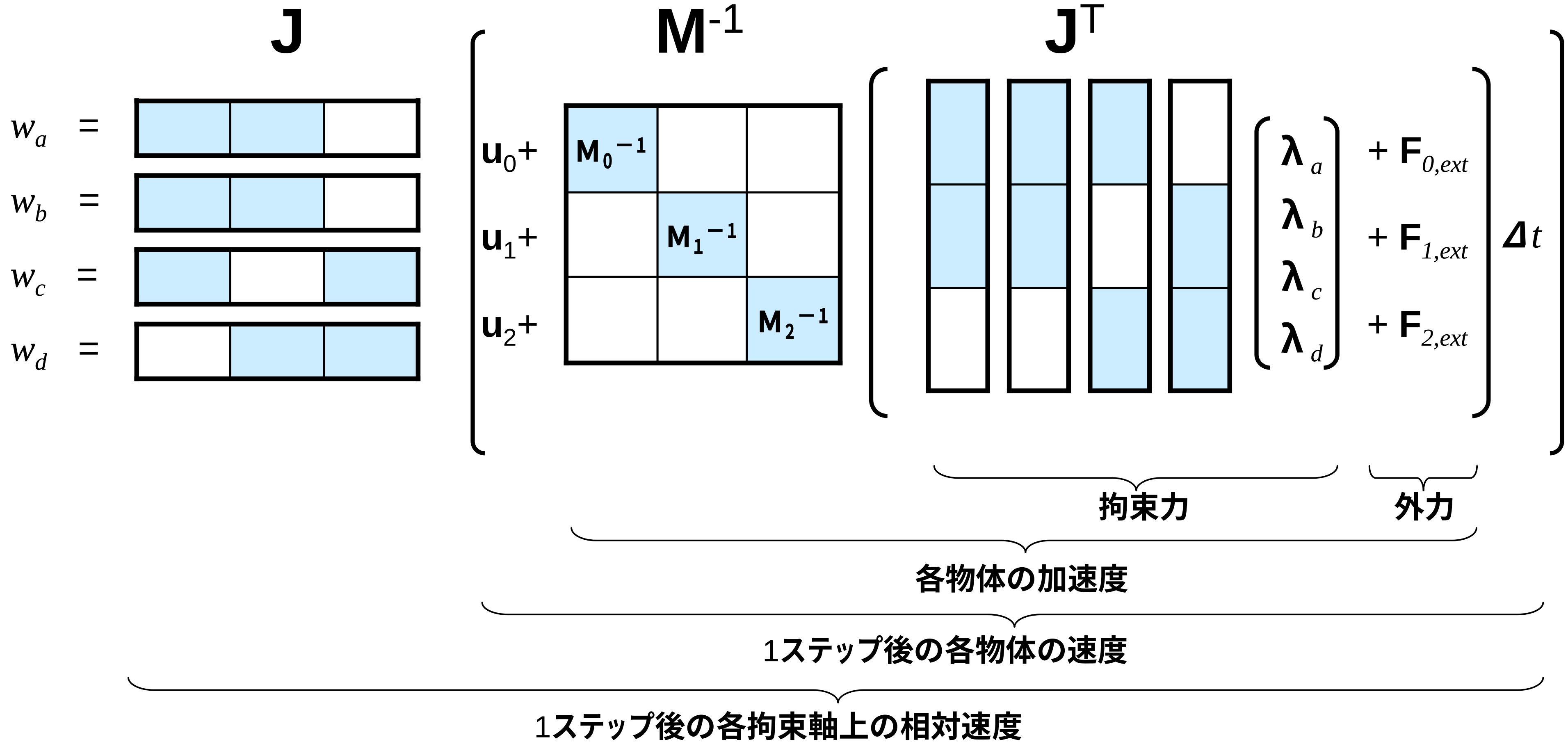
拘束の数

	物体0	物体1	物体2
$\mathbf{J}_a =$	$\mathbf{J}_{a,0}$	$\mathbf{J}_{a,1}$	
$\mathbf{J}_b =$	$\mathbf{J}_{b,0}$	$\mathbf{J}_{b,1}$	
$\mathbf{J}_c =$	$\mathbf{J}_{c,0}$		$\mathbf{J}_{c,2}$
$\mathbf{J}_d =$		$\mathbf{J}_{d,1}$	$\mathbf{J}_{d,2}$

多体の物理 (拘束力)

$$\begin{array}{lc} \text{物体0への拘束力} & = \\ \text{物体1への拘束力} & = \\ \text{物体2への拘束力} & = \end{array}
\begin{array}{c} \mathbf{J}_a^{T} \\ \begin{bmatrix} \mathbf{J}_{a,0}^{T} \\ \mathbf{J}_{a,1}^{T} \\ \ \end{bmatrix} \end{array} \times \boldsymbol{\lambda}_a +
\begin{array}{c} \mathbf{J}_b^{T} \\ \begin{bmatrix} \mathbf{J}_{b,0}^{T} \\ \mathbf{J}_{b,1}^{T} \\ \ \end{bmatrix} \end{array} \times \boldsymbol{\lambda}_b +
\begin{array}{c} \mathbf{J}_c^{T} \\ \begin{bmatrix} \mathbf{J}_{c,0}^{T} \\ \ \\ \mathbf{J}_{c,2}^{T} \end{bmatrix} \end{array} \times \boldsymbol{\lambda}_c +
\begin{array}{c} \mathbf{J}_d^{T} \\ \begin{bmatrix} \ \\ \mathbf{J}_{d,1}^{T} \\ \mathbf{J}_{d,2}^{T} \end{bmatrix} \end{array} \times \boldsymbol{\lambda}_d$$

多体の物理 (拘束条件式)


J
M-1
JT
wa =
wb =
wc =
wd =
u0+
u1+
u2+
M0−1
M1−1
M2−1
λa
λb
λc
λd
+ F0,ext
+ F1,ext
+ F2,ext
Δt
拘束力
外力
各物体の加速度
1ステップ後の各物体の速度
1ステップ後の各拘束軸上の相対速度

多体の物理 (拘束条件式)

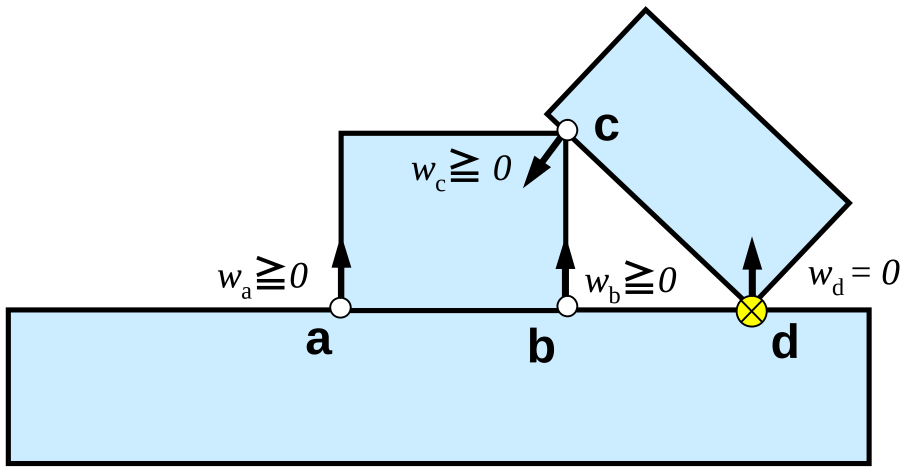
$$\begin{pmatrix} w_a \\ w_b \\ w_c \\ w_d \end{pmatrix} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\boldsymbol{\lambda}\,\Delta t + \mathbf{J}(\mathbf{u} + \mathbf{M}^{-1}\mathbf{F}_{ext}\,\Delta t) = \mathbf{A}\boldsymbol{\lambda} + \mathbf{b}$$

(簡単のために 最右辺で Δt は λ に含めた)

多体の物理 (拘束条件式)

結局ここまでで何が定式化できたかというと…

$$\mathbf{A}\boldsymbol{\lambda} + \mathbf{b} = \begin{array}{lcl} w_a & \geqq & 0 \\ w_b & \geqq & 0 \\ w_c & \geqq & 0 \\ w_d & = & 0 \end{array}$$

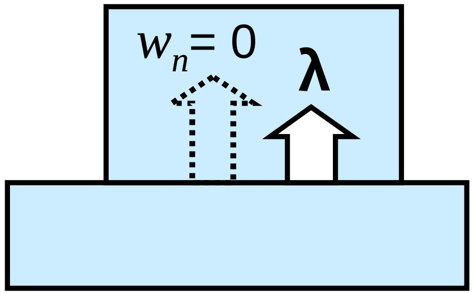


方程式と不等式が混在！

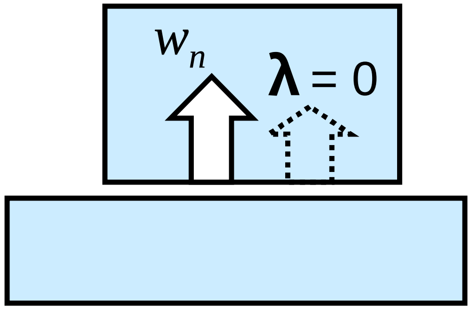
LCP (線形相補問題)

不等式拘束は一見厄介に見えるが、
上方向の相対速度 w_n と拘束力 λ には以下の関係があることに着目する

λ > 0 のとき $w_n = 0$
(2物体が接触している間は力が働く)



λ = 0 のとき $w_n > 0$
(2物体が離れはじめたら力は働かない)

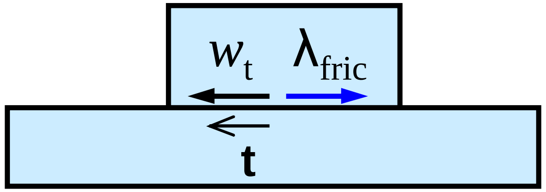


このような問題を
線形相補問題(Linear Complementary Problem)と呼ぶ

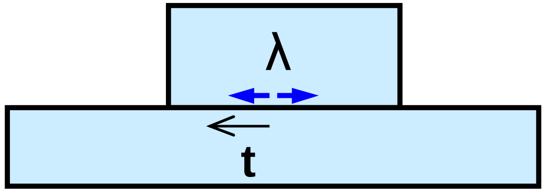
LCP (摩擦の場合)

軸 t 上での2物体の相対速度を w_t、最大摩擦力を $\boldsymbol{\lambda}_{fric}$とすると

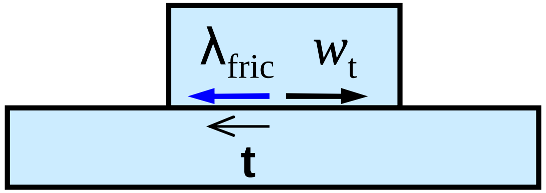
$\boldsymbol{\lambda} = -\boldsymbol{\lambda}_{fric}$ のとき $w_t > 0$



$-\boldsymbol{\lambda}_{fric} < \boldsymbol{\lambda} < \boldsymbol{\lambda}_{fric}$ のとき $w_t = 0$



$\boldsymbol{\lambda} = \boldsymbol{\lambda}_{fric}$ のとき $w_t < 0$



LCP の一般形

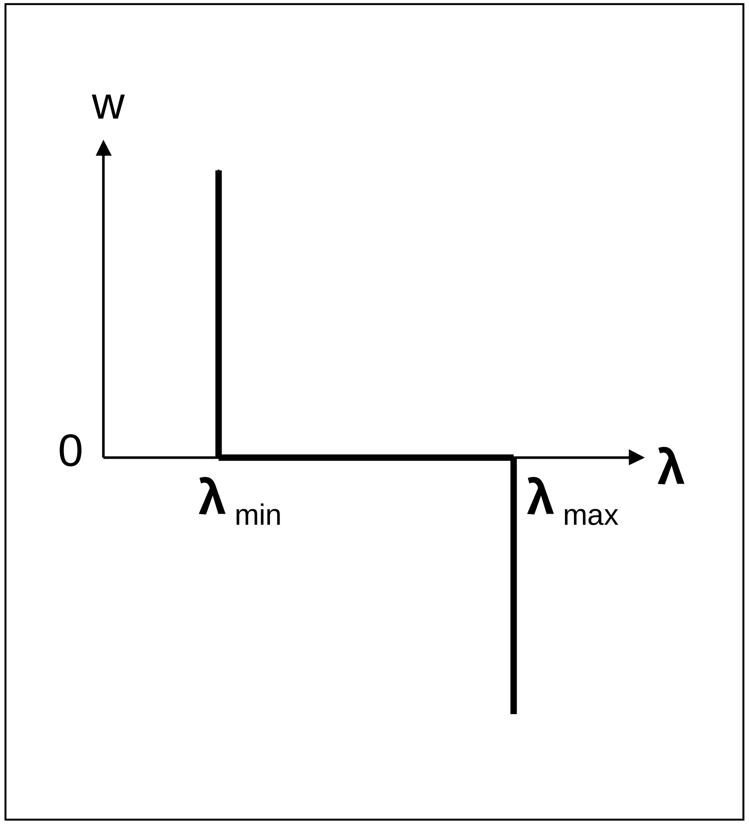
$w = A\boldsymbol{\lambda} + b$ とする

以下を満たす w と $\boldsymbol{\lambda}$ を求める

$\boldsymbol{\lambda} = \boldsymbol{\lambda}_{min}$ のとき $w > 0$

$\boldsymbol{\lambda}_{min} < \boldsymbol{\lambda} < \boldsymbol{\lambda}_{max}$ のとき $w = 0$

$\boldsymbol{\lambda} = \boldsymbol{\lambda}_{max}$ のとき $w < 0$



LCP の一般形

ＬＣＰの一般形でさまざまな拘束が統一的に表現できる
(赤字部分だけが有効)

貫通なし拘束 ($\lambda_{min} = 0,\ \lambda_{max} = \infty$)

$\lambda = 0$ のとき $w > 0$
$0 < \lambda < \infty$ のとき $w = 0$
$\lambda = \infty$ のとき $w < 0$

等式拘束 ($\lambda_{min} = -\infty,\ \lambda_{max} = \infty$)

$\lambda = -\infty$ のとき $w > 0$
$-\infty < \lambda < \infty$ のとき $w = 0$
$\lambda = \infty$ のとき $w < 0$

LCP の解き方

- 直接法
  - Lemke-Howson のアルゴリズム[Eberly04]

- 繰り返し法
  - 実用的なエンジンでは普通こちらを採用
  - 通常の連立方程式の繰り返し解法とほぼ同じ解き方
  - 唯一の違いは「クランプ」処理

Gauss-Seidel 法

- 線形方程式の解法
- 非常に簡単な方法でありながらかなり実用的
- 繰り返しによって解を収束させる
- 不定の場合は平均的な解を返す傾向

Gauss-Seidel 法

線形方程式を以下のように変形

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

$$\begin{bmatrix} -a_{00} & 0 & 0 \\ 0 & -a_{11} & 0 \\ 0 & 0 & -a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & a_{01} & a_{02} \\ a_{10} & 0 & a_{12} \\ a_{20} & a_{21} & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & -a_{01}/a_{00} & -a_{02}/a_{00} \\ -a_{10}/a_{11} & 0 & -a_{12}/a_{11} \\ -a_{20}/a_{22} & -a_{21}/a_{22} & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_0/a_{00} \\ b_1/a_{11} \\ b_2/a_{22} \end{bmatrix}$$

Gauss-Seidel 法

変形した方程式を「更新式」と解釈

k 回目から $k+1$ 回目の更新は

$$\begin{pmatrix} x_{0,k+1} \\ x_{1,k+1} \\ x_{2,k+1} \end{pmatrix} = \mathbf{A} \begin{pmatrix} x_{0,k} \\ x_{1,k} \\ x_{2,k} \end{pmatrix} + \mathbf{b}$$

更新を繰り返し $\mathbf{x}_{k+1} = \mathbf{x}_k = \mathbf{x}_{conv}$ になったとする

この $\mathbf{x}_{conv}$ は元の方程式を満たすので求める解となる

（方程式から更新式への変形を逆にたどれば明らか）

Gauss-Seidel 法

i 行目の更新に i - 1 行目までの更新結果を即時利用

$$\boxed{x_{0,k+1}} = a_{00}\, x_{0,k} + a_{01} x_{1,k} + a_{02} x_{2,k} + b_0$$

$$\boxed{x_{1,k+1}} = a_{00} \boxed{x_{0,\mathbf{k+1}}} + a_{01} x_{1,k} + a_{02} x_{2,k} + b_1$$

$$x_{2,k+1} = a_{00} \boxed{x_{0,\mathbf{k+1}}} + a_{01} \boxed{x_{1,\mathbf{k+1}}} + a_{02} x_{2,k} + b_2$$

即時利用しない場合 (ヤコビ法) に比べて

1．5～2倍程度，収束速度が向上

投影型 Gauss-Seidel 法

ＬＣＰを解く効率的な方法

繰り返しのたびにλをクランプ

$$\boldsymbol{\lambda} = \mathrm{clamp}(\boldsymbol{\lambda}, \boldsymbol{\lambda}_{min}, \boldsymbol{\lambda}_{max})$$

たったこれだけ！

投影型 Gauss-Seidel 法

なぜこれで ＬＣＰ が解ける？

$\mathbf{w} = \mathbf{A}\boldsymbol{\lambda} + \mathbf{b}$ の A をよく調べてみる

$\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}$ ならばその対角成分 a_{kk} は必ず正値

投影型 Gauss-Seidel 法

A の対角成分 a_{kk} **は必ず正値**

w = **A λ** + **b の** k **行目を取り出してみると**

$$w_k = a_{k1}\lambda_1 + a_{k2}\lambda_2 + \ldots + a_{kk}\lambda_k + \ldots + a_{kn-1}\lambda_{n-1} + a_{kn}\lambda_n + b_k$$

$\lambda_{min} < \lambda_k < \lambda_{max}$ **のとき**

w_k **は A λ + b = 0 という方程式の** k **行目であるから** 0 **に収束**

$\lambda_k < \lambda_{min}$ **のとき** $\lambda_k \leftarrow \lambda_{min}$ **とすると**

w_k **はクランプ前より大きな値になる。つまり** $w_k > 0$ **に収束**

$\lambda_k > \lambda_{max}$ **のとき** $\lambda_k \leftarrow \lambda_{max}$ **とすると**

w_k **はクランプ前より小さな値になる。つまり** $w_k < 0$ **に収束**

安定化と高速化

安定化と高速化

- 安定性とは
  - 力が釣り合っていれば静止または等速運動すべき
- 安定化と高速化は表裏一体
  - 早く安定すればソルバの繰り返し数は少なくてすむ
- スリープ機能は安定化には寄与しない
  - 安定したからスリープにできるだけ

安定性が要求される場面

- スタッキング (積み重ね)
  - 安定性が低いと崩壊
- ラグドールなど複雑なジョイントの相互作用
  - 安定性が低いといつまでも「死なない」

安定化のテクニック

- Slop(許容貫通誤差)
- Permutation(行入れ替え)
- WarmStart (前ステップ解による初期化)
- Shock Propagation (スタッキング対策)

安定化のテクニック

- WA法(ShockPropagationの改良手法)
- ＡＳ法 (スリープの改良手法)

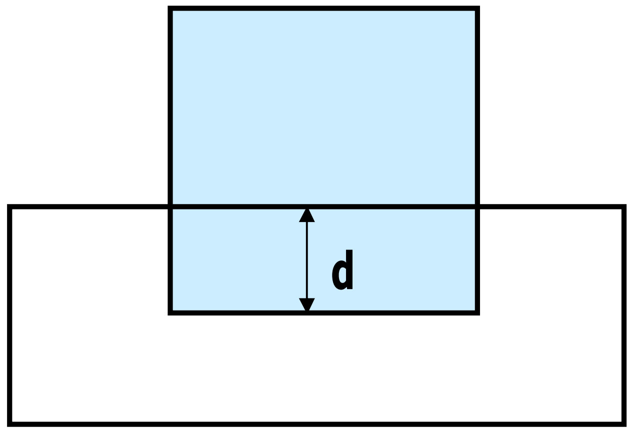
貫通の解消

貫通してしまった場合は貫通を解消するための速度を上乗せする
貫通量を d, 1 ステップ後の相対速度を u' とする
拘束条件式を

$$\mathbf{J}u' \geqq v_{error}$$

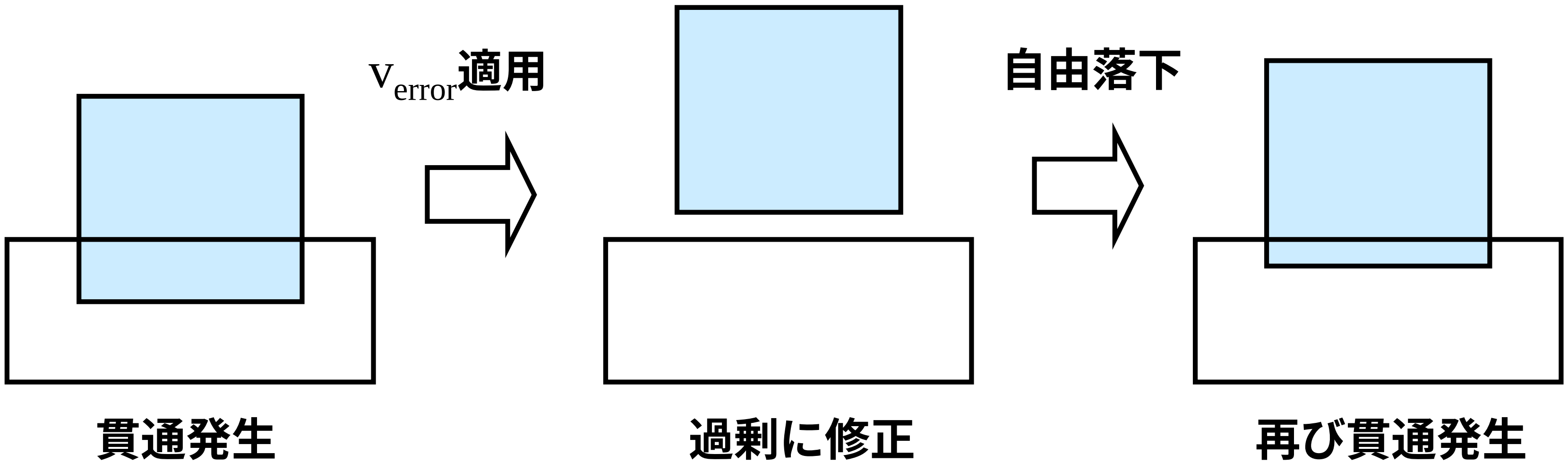
$$v_{error} = kd / \Delta t$$ (kは1以下の適当な低減係数)

とすればよい



Slop (許容貫通誤差)

単純に貫通解消速度 v_{error} を条件式に含めると振動が発生する [Catt08]

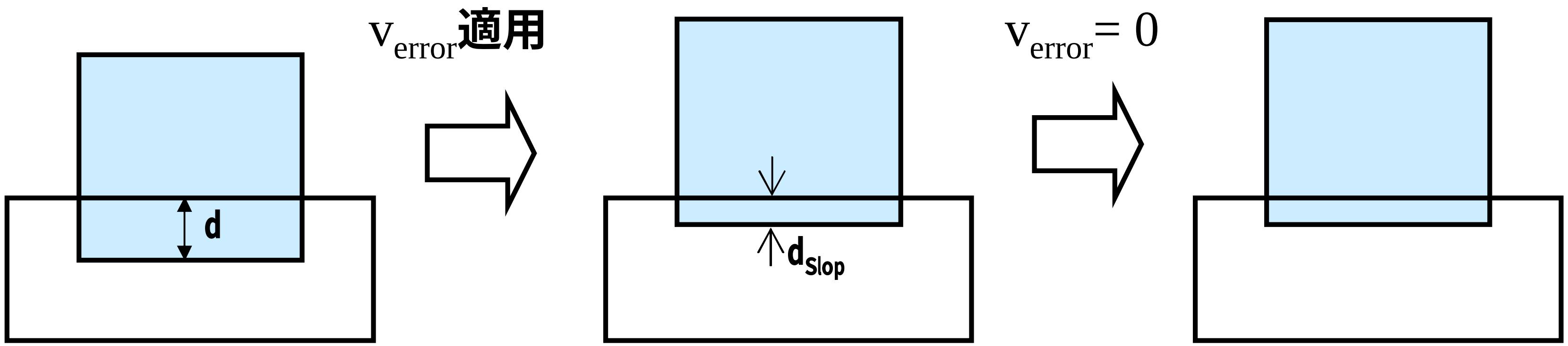


Slop (許容貫通誤差)

完全に貫通を解消してはいけない

深さ d_{slop} だけ貫通を残す → 支持力を安定的に発生させる

$$v_{error} = (d - d_{slop})/\Delta t$$



貫通発生

d_{Slop}だけ貫通を残す

わずかに貫通した状態で安定

Permutation(行入れ替え)

- LCPの各行は拘束に対応
- 拘束行列は積み上げた順番を反映しやすい
- 積み上げた順番に解くと誤差が集積
  - シミュレーションが不安定化
- 各行をランダムに入れ替えた後ソルバを起動

WarmStart
(前ステップの解による初期化)

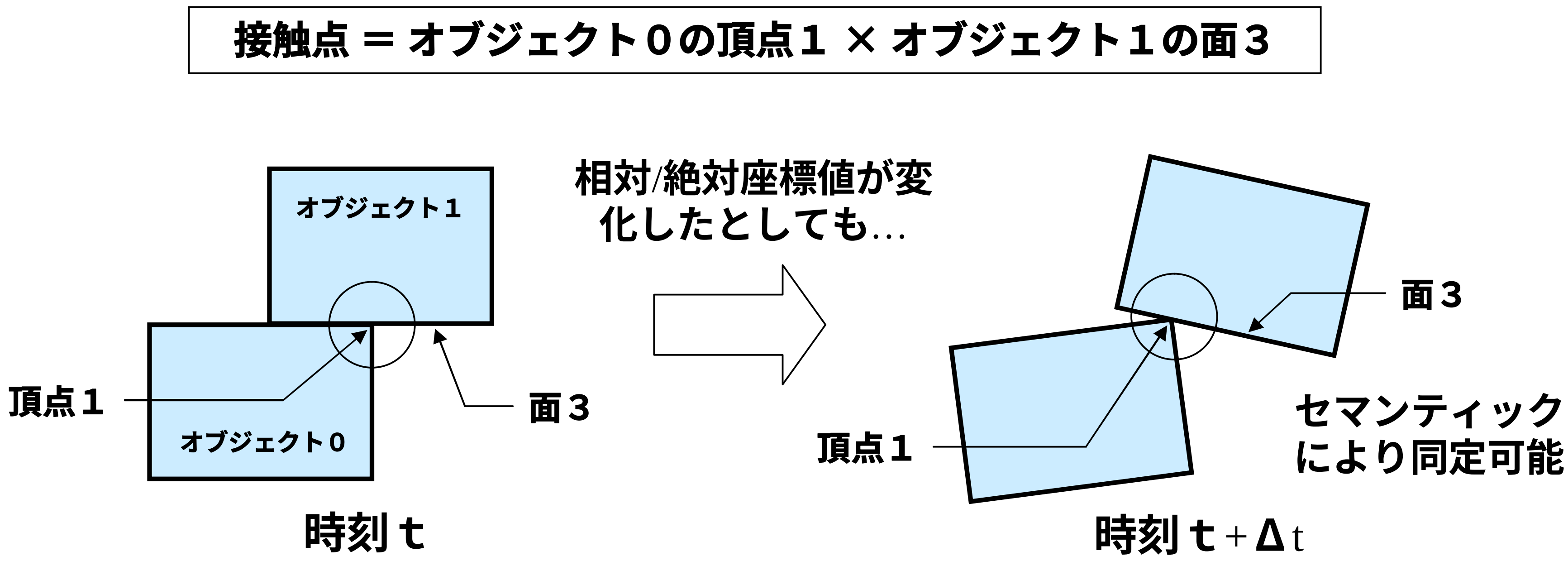
- Gauss-Seidel 法は繰り返しによる収束計算
- 初期値は解に近いほどよい
- 1ステップ前の結果を初期値とする
- 安定化/高速化の効果は絶大

WarmStart

- 前ステップの解とは拘束力λ
- 新ステップで前回のλをどの点に適用するか
- ジョイントは固定的に存在
  ⇒ 自明
- 接触点はダイナミックに変化
  ⇒ 同じ接触点を探す必要がある

WarmStart

- 接触点の生成状況はステップ間で完全に同じではない
- 接触点は何らかのセマンティックでID化しておく [Moravan04]

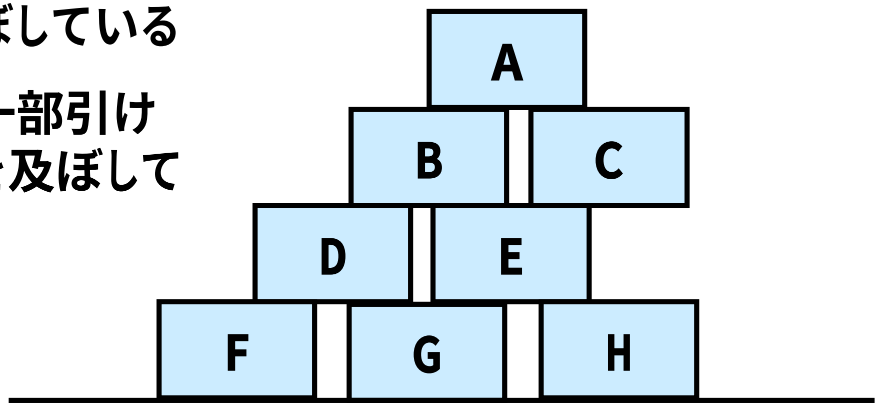


スタッキング（積み重ね）

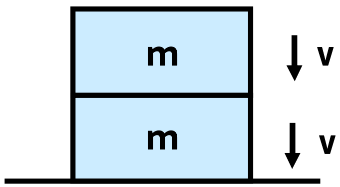
- スタッキングの処理はなぜ難しいのか？
  - 多段階の力の伝播をシミュレーションする必要あり
  - 段数が多いほど繰り返しが多く必要

GにはA、B，C，D、Eが力を及ぼしている

さらにFはDの、HはEの加重を一部引け受けており、間接的にGに影響を及ぼしている



スタッキング
(インパルス法による解法)

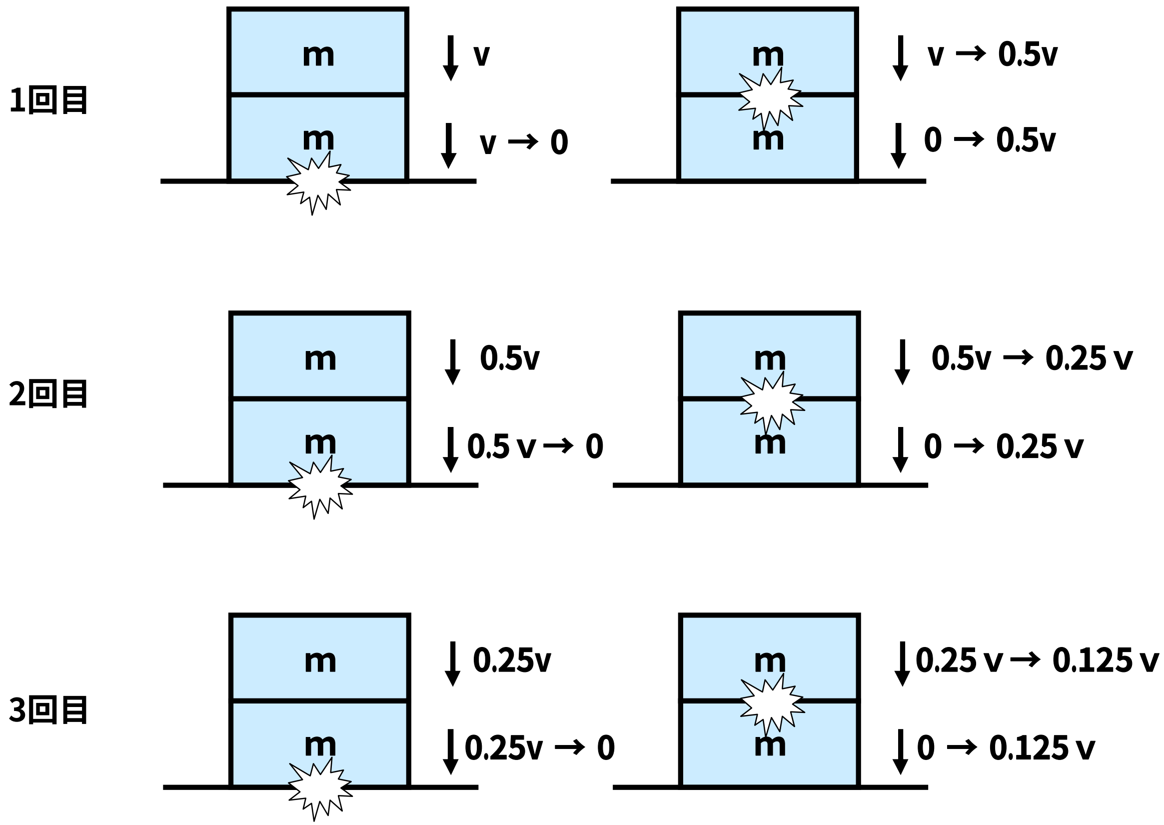


運動量保存則

$m_0\mathbf{v'_0} + m_1\mathbf{v'_1} = m_0\mathbf{v_0} + m_1\mathbf{v_1}$

２物体間の衝突を繰り返し計算し力を伝播させる

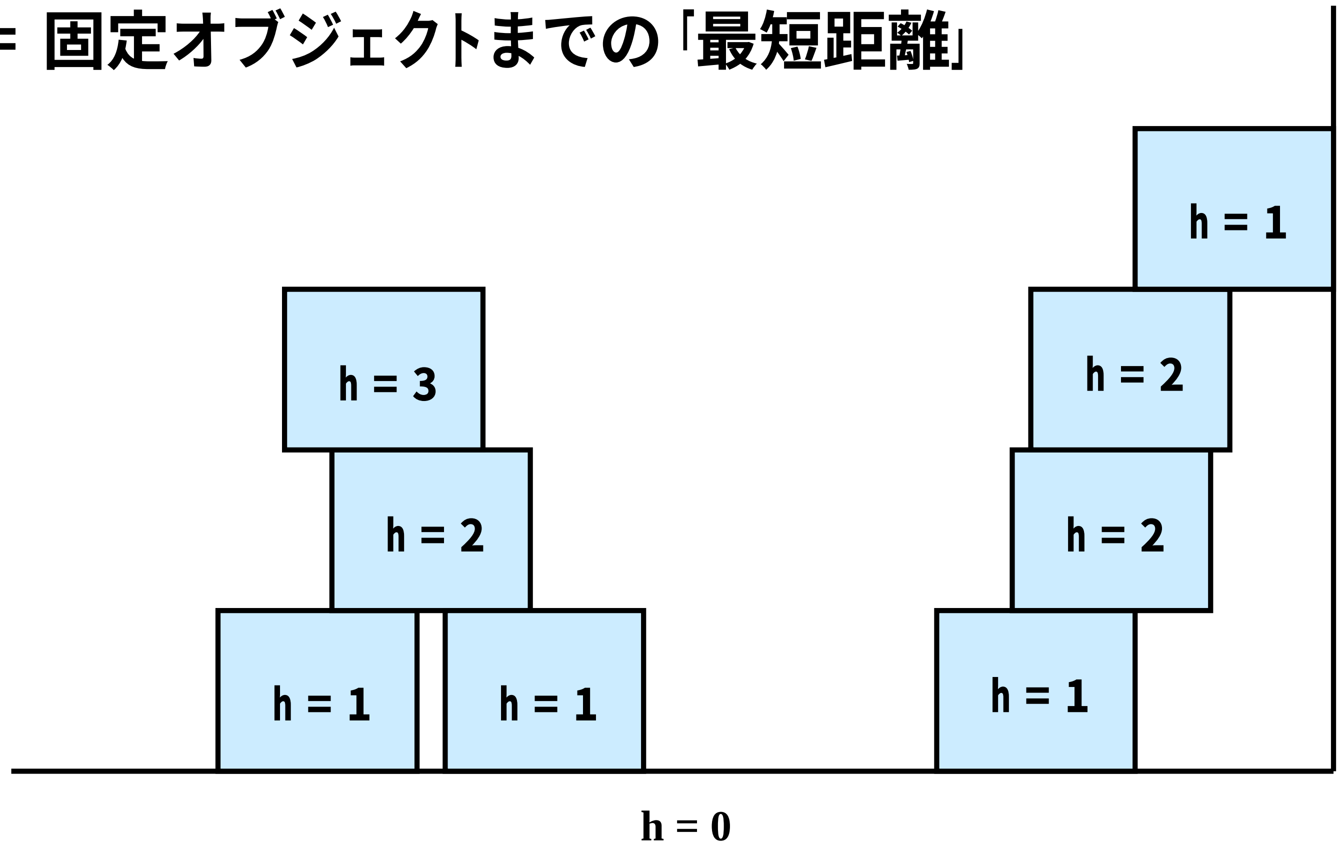
解析法（ＬＣＰ）の繰り返し解法も原理的にはこれと同じ



反発係数はすべて０とする

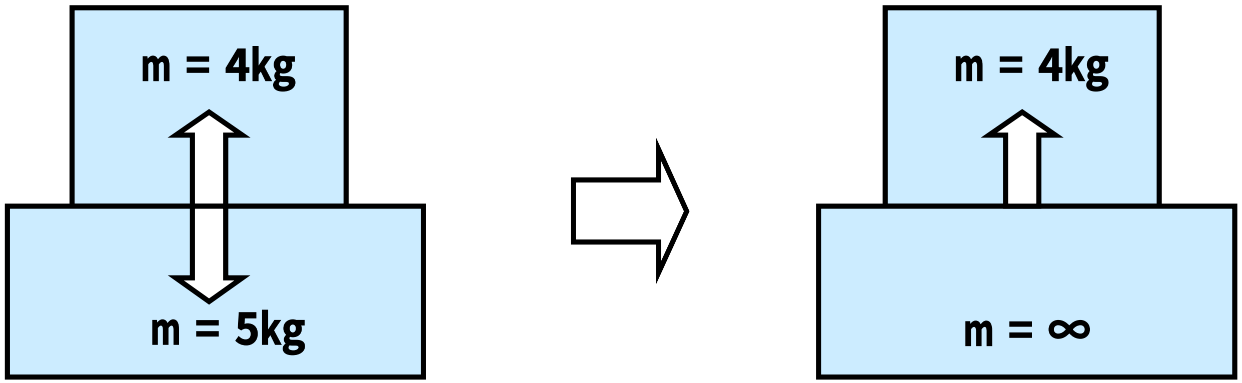
物体の「高さ」

- 積み重なった物体の「上」「下」とは？
- 高さ h ＝ 固定オブジェクトまでの「最短距離」



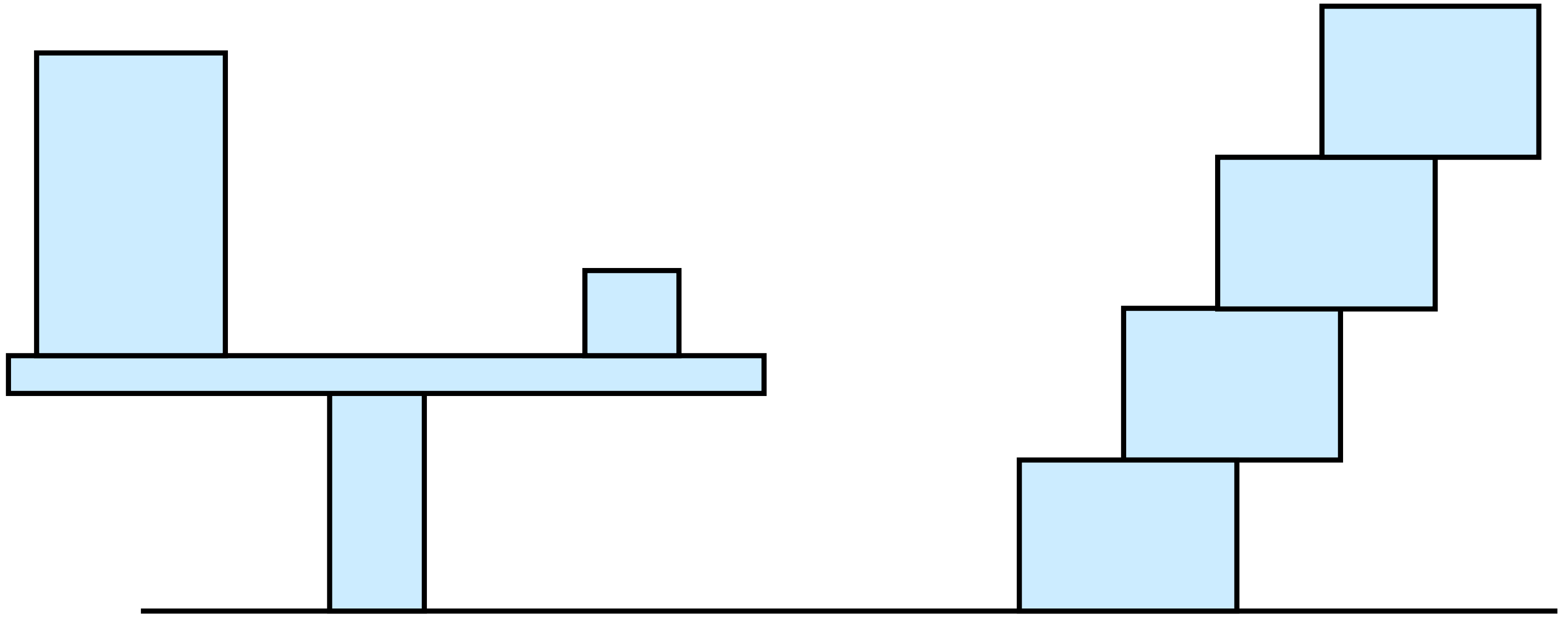
Shock Propagation

- [Guendelman03]
- スタッキング問題への対策（ある種の「フェイク」）
- 「下側」オブジェクトの質量を強制的に∞とする
- 力の伝播方向を「下」から「上」に制限



WeighFeeling (重さ感知) 問題

- ShockPropagationの副作用
- 過度に安定してしまう



WeightFeeling (重さ感知) 問題

とりあえずの解決策

λ_0 = 上下の質量を変更しない場合の解

λ_1 = 上下の質量を m：∞ とした場合の解

2つの解を加重平均

$\lambda = (1 - c)\lambda_0 + c\lambda_1$

しかし…

ShockPropagationの冗長さ

- LCPに適用した場合2回のシミュレーションが必要
- ソルバのコストは2倍

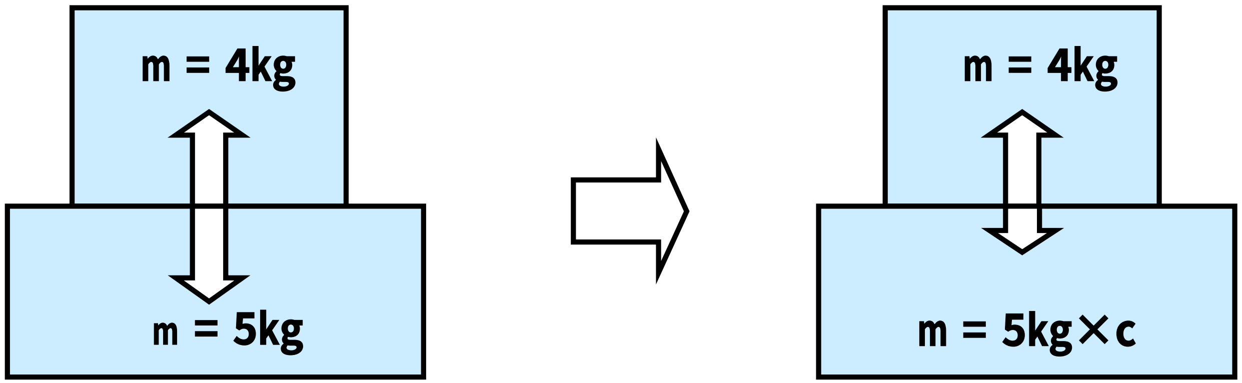
ShockPropagationの冗長さ

- 「まず風呂を100度に沸かす」
- 「熱すぎるので水でうめて40度にしました」
- 何か変？

- 最初から40度で沸かせばいいのでは…

WA法 (Weight Amplification : 重さ増幅)

- 「下側」のオブジェクトに増幅係数 c をかける
- $c = 1.4$ 程度でかなりよい結果
- 数十段のスタックでも処理可能
- WeightFeeling 問題もノーコストで解決

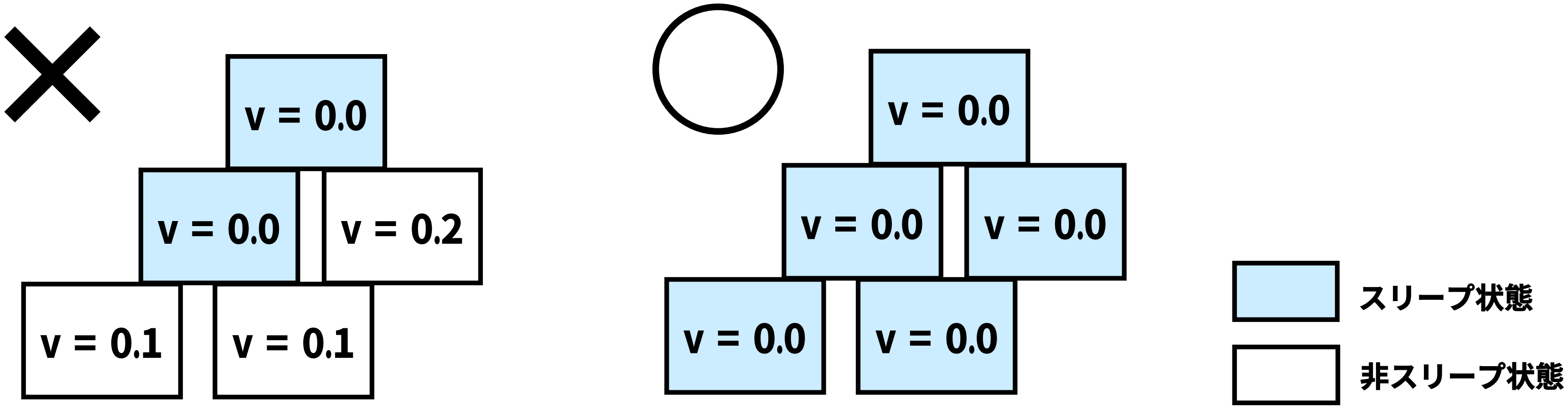


スリープ

- 速度がほぼ０ならシミュレーションから除外
- 無駄なシミュレーションコストの削減

スリープ

- 「シミュレーションアイランド」レベルで適用
- オブジェクト単独ではスリープにはできない
- スリープはシミュレーションの安定化には寄与しない



ＡＳ法 (Aggressive Sleep : 積極的スリープ)

Ｓlop

接触点の生成状況が安定化

↓

解くべきＬＣＰの構造が安定化

Aggressive Sleep

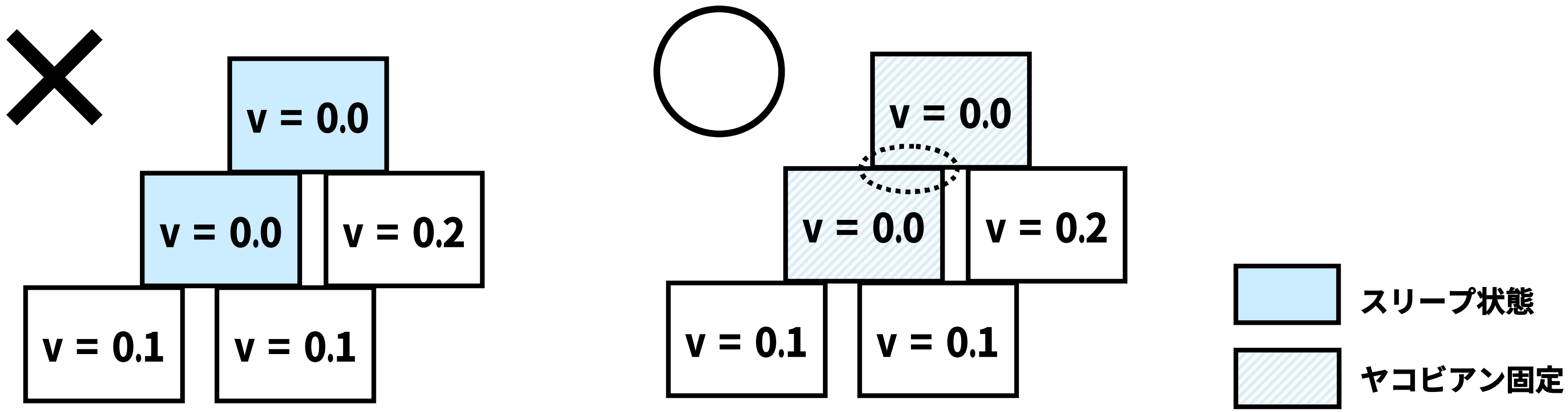
静止したオブジェクト間のヤコビアンを固定 (前回の値を使用)

↓

解くべきLCPの構造が安定化

ＡＳ法 (Aggressive Sleep : 積極的スリープ)

- ヤコビアンは前ステップの値を使用 (高速化にも貢献)
- オブジェクト単位で適用可能
- 静止の判定は粗くてＯＫ (調整が簡単)
- シミュレーションは行うので不正な静止は起こらない



参考文献

[Baraff 89] David Baraff. Analytical method for dynamics simulation of nonpenetrating bodies. *Computer Graphics Vol.23, No.3, 223-232, 1989.*

[Catt08] Erin Catto. Modeling and Solving constraints. *GDC2008 Tutorial Note.*

[Eberly04] David H. Eberly. *Game Physics, Morgan Kaufmann Publishers.*

[Erleben05] Kenny Erleben. Stable,robust,and versatile multibody dynamics animation. *PhD.,thesis, Department of Computer Science, University of Copenhagen,Denmark,2005.*

[Erleben07] Kenny Erleben. Velocity-Based Shock Propagation for Multibody Dynamics Animation. *ACM Transactions on Graphics, Vol.26, No.2, June 2007.*

[Garst03] Helmut Garstenauer. A Unified Framework for Rigid Body Dynamics. *Master thesis, 2003.*

[Guendelman03]Eran Guendelman et al. Nonconvex rigid bodies with stacking. *ACM Tansaction on Graphics, Vol.22 Issue 3, July 2003.*

[Moravan04]Adam Moravanszky et al. Fast Contact Reduction for Dynamics Simulation. *Game Programming Gems4, Charles River Media.*