

# 剛体シミュレーション Rigid Body Simulation

大阪大学 大学院基礎工学研究科

機能創成専攻 生体工学領域 大城研究室

吉元俊輔

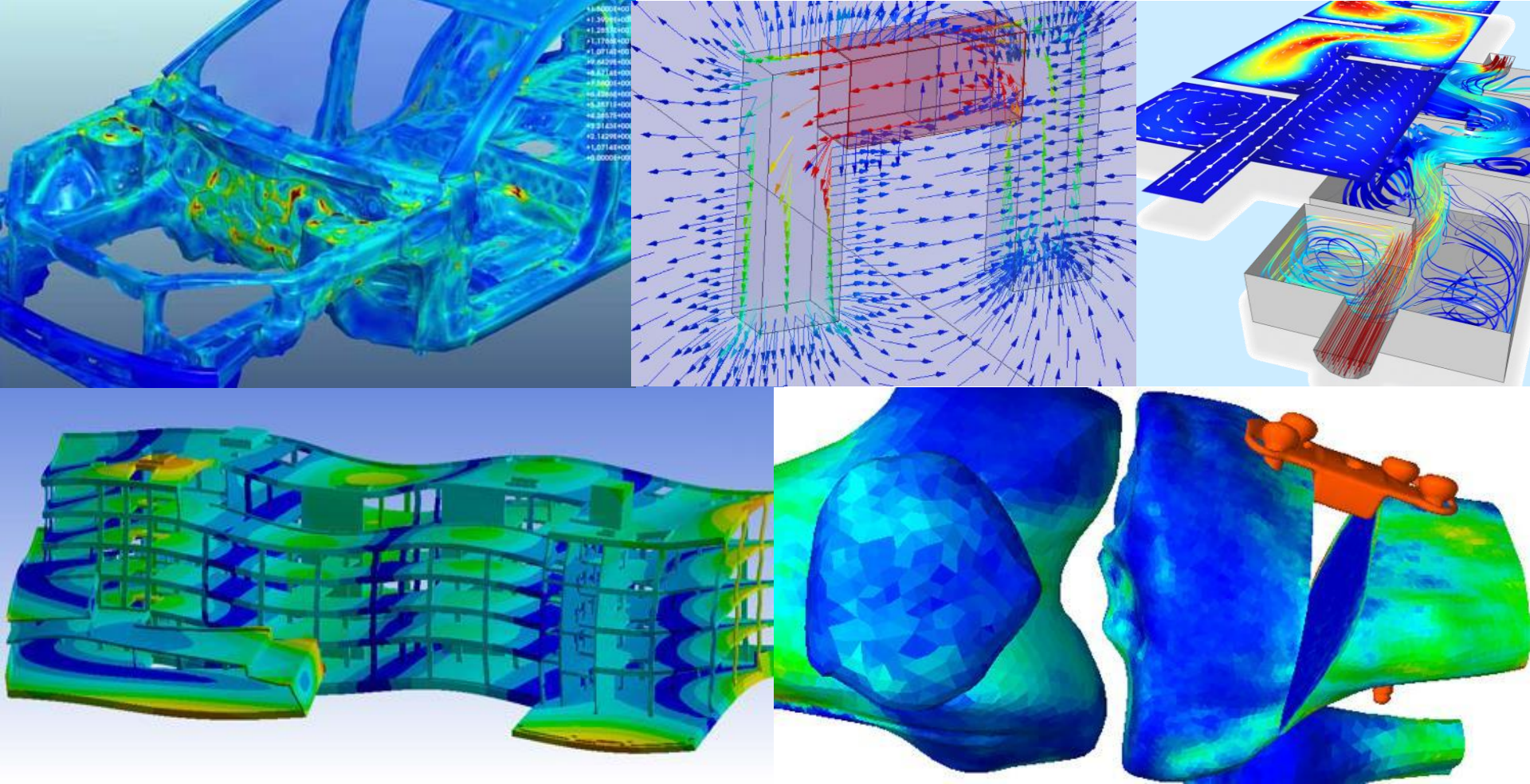
平成29年4月21日（金）

物理シミュレーション

**デモ**

物理シミュレーション

# 基礎編



物理シミュレーションは様々な分野で有用な技術

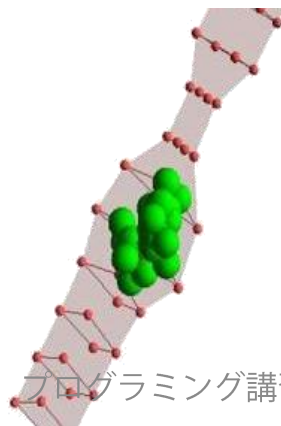
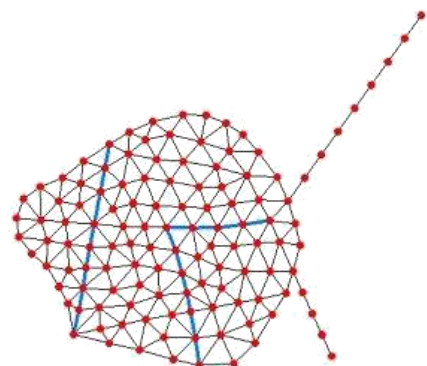
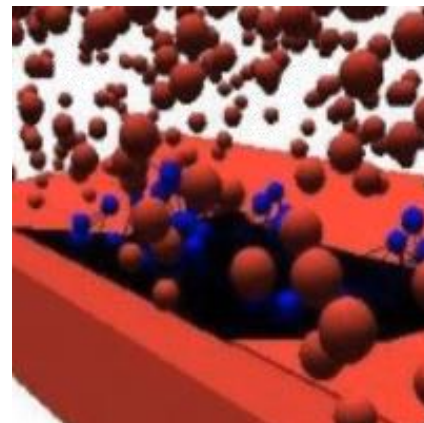
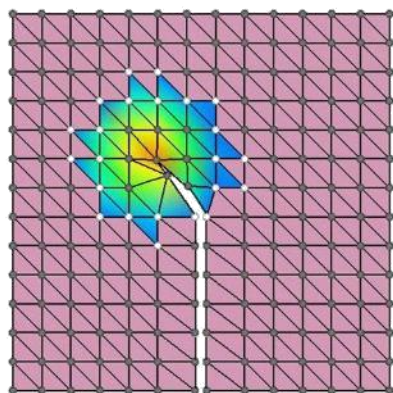
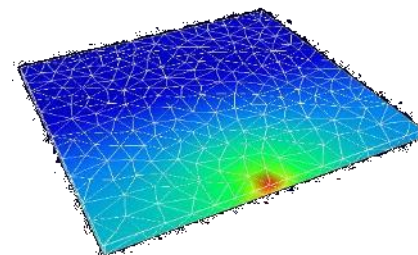
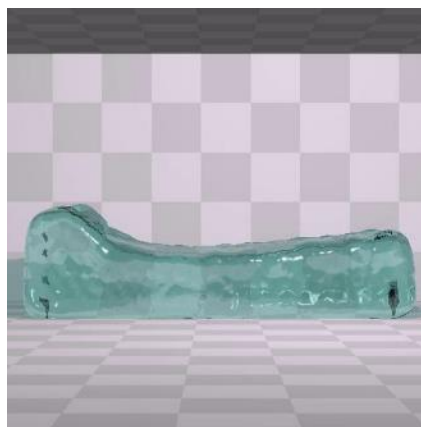
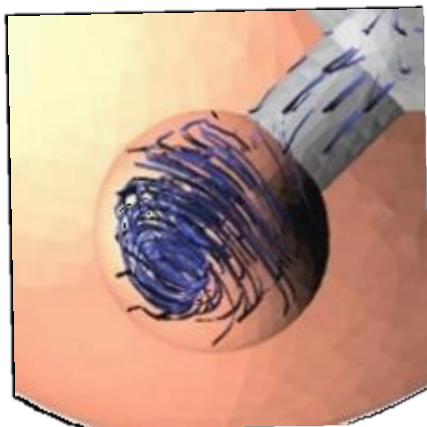
# そもそもシミュレーションって

## 「シミュレーション」の元々の意味は「模擬実験」

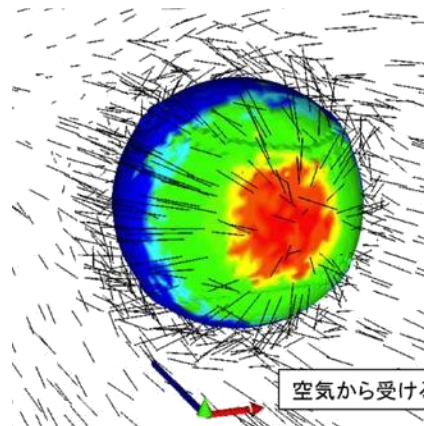
現象のメカニズムを解明するために、実際の現象に類似した現象を条件の制御が可能な実験で再現させ、種々の計測を行うこと

- 現象を事前に理解することができ必要な対策を講じることにも可能
- 実物を使った実験をしなくて済むのでコストや手間を減らせる
- 実験が難しい現象や観測が困難な事象を解析できる
- 結果が数値でわかるので精密な議論ができる
- 自然界では起こりえない現象を調べることができる





プログラミング講習会



空気から受ける

# 物理シミュレーション： 支配方程式・原理の数値解を得る

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{1}{\rho} \nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{F}$$

$$\nabla \cdot \mathbf{B} = 0$$

$$C_v \frac{\partial T}{\partial t} = -\lambda \frac{\partial^2 T}{\partial x^2}$$

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0$$

$$\frac{1}{s^2} \frac{\partial^2 u}{\partial t^2} = \nabla^2 u$$

$$\nabla \cdot \mathbf{D} = \rho$$

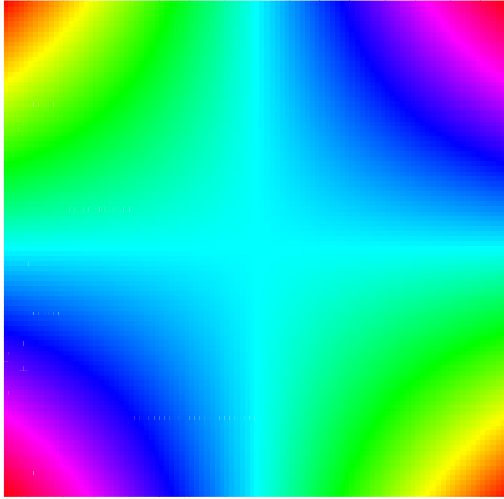
$$\mathbf{F} = m \frac{d^2 \mathbf{x}}{dt^2} + c \frac{d\mathbf{x}}{dt} + k\mathbf{x}$$

$$\nabla \times \mathbf{H} - \frac{\partial \mathbf{D}}{\partial t} = \mathbf{j}$$

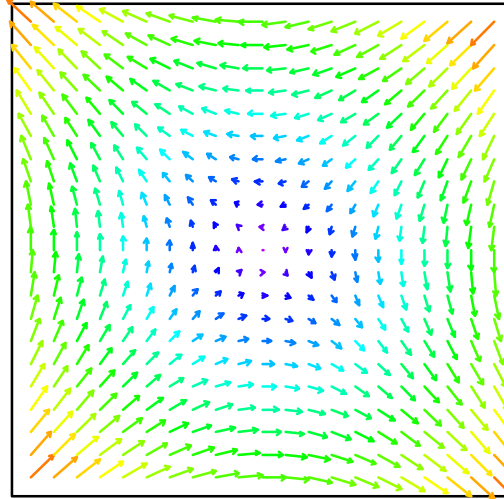
$$\frac{\partial \phi}{\partial t} = D \nabla^2 \phi$$

$$\int_v \{\epsilon\}^T \{\sigma\} dv - \int_v \{U\}^T \{\bar{G}\} dv - \int_{S_\sigma} \{U\}^T \{\bar{T}\} ds = 0$$

# スカラー場？ベクトル場？微分オペレータ？



スカラー場



ベクトル場

$$\nabla = \left\{ \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right\}$$

空間微分

$$\nabla f = \left\{ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right\} = \text{grad} f$$

$$\nabla \times \mathbf{v} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ v_x & v_y & v_z \end{vmatrix} = \text{rot} \mathbf{v}$$

$$\nabla \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = \text{div} \mathbf{v}$$

$$\Delta f = \nabla^2 f$$

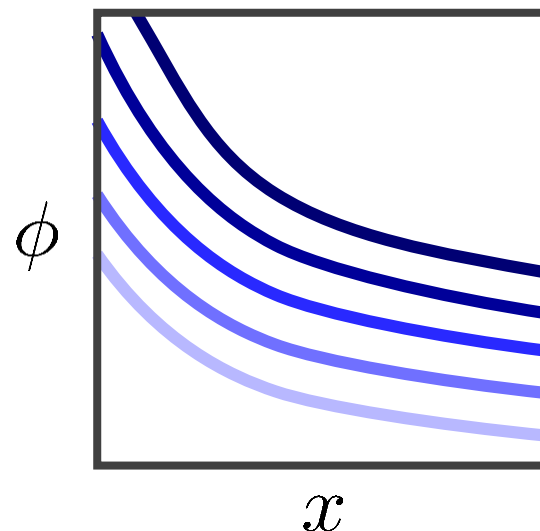


# 初期条件・境界条件を与えて解く

境界条件がないとき

$$\frac{d\phi(x)}{dx} = -\phi(x)$$

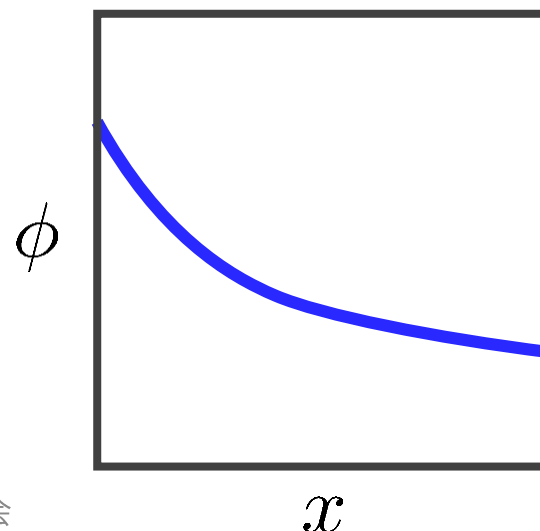
$$\phi(x) = A \exp(-x)$$



境界条件があるとき

$$\frac{d\phi(x)}{dx} = -\phi(x) \quad \boxed{\phi(0) = 1}$$

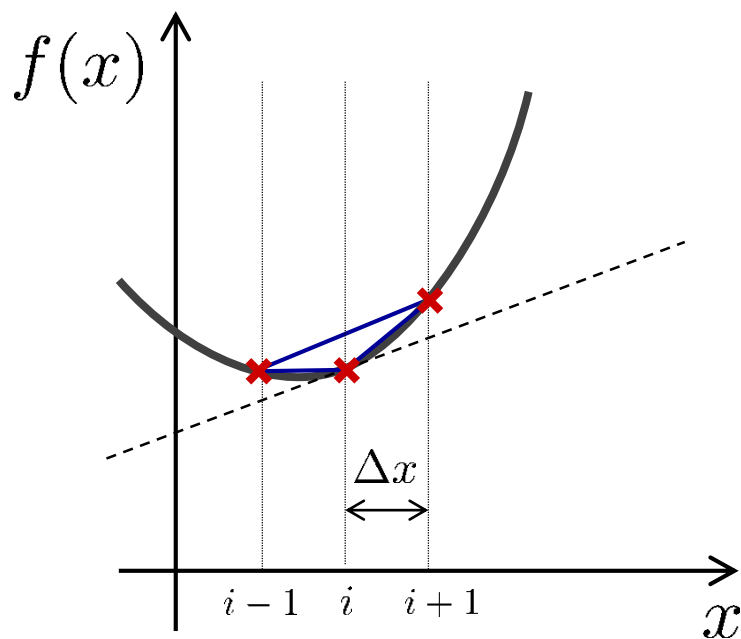
$$\phi(x) = \exp(-x)$$



# 離散化：コンピュータで扱える形に

$$f(x) \xrightarrow{\text{離散化}} f[i]$$

$$\frac{df(x)}{dx} \xrightarrow{\text{離散化}}$$



前進差分

$$\frac{f[i+1] - f[i]}{\Delta x}$$

中心差分

$$\frac{f[i+1] - f[i-1]}{2\Delta x}$$

後退差分

$$\frac{f[i] - f[i-1]}{\Delta x}$$

# 数値解析で押さえておきたいキーワード

- 誤差、正確さ、安定性
- ニュートン法、2分法
- ガウスの消去法、反復法
- 多項式補間、最小二乗法
- 区分求積法、台形公式、シンプソンの公式
- オイラー法、クランク・ニコルソン法、ルンゲ・クッタ法
- ラグランジュの未定乗数法

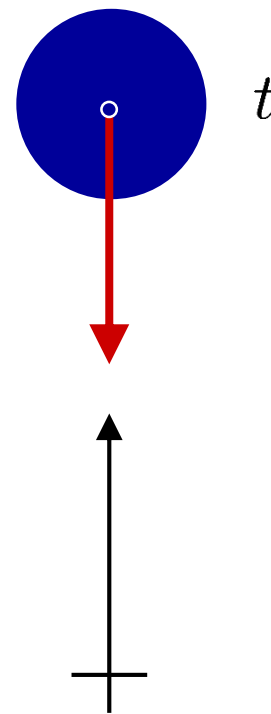
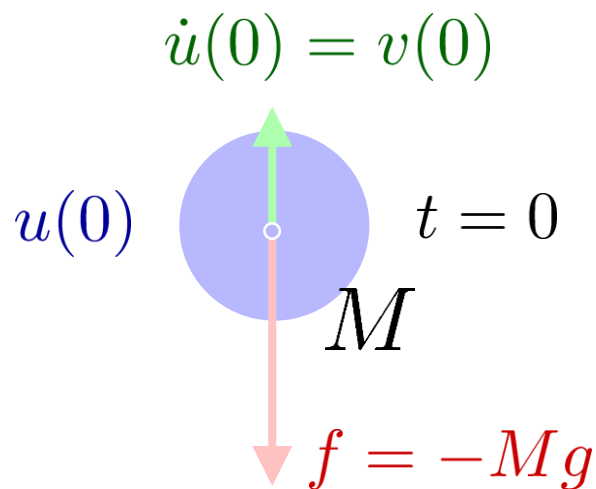
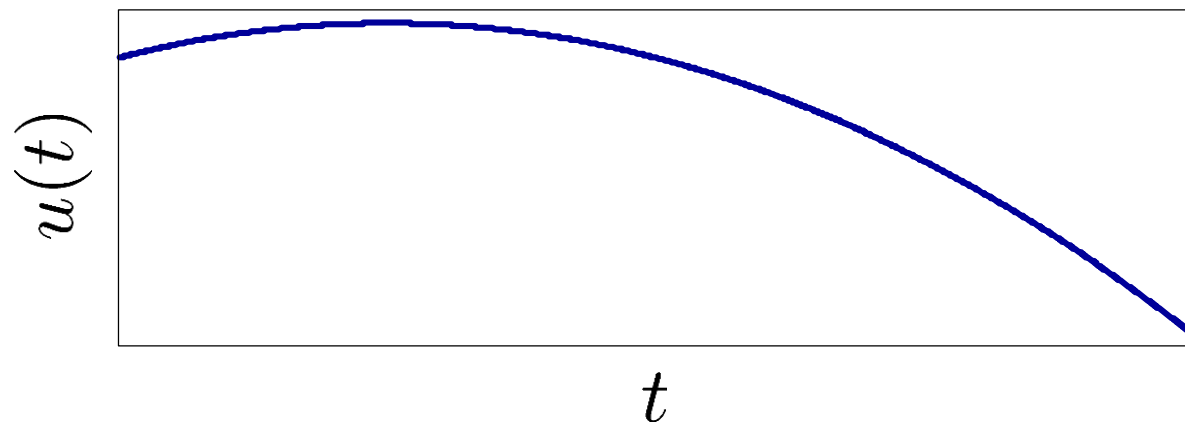
物理シミュレーション

# 剛体の運動

# 運動方程式：1次元の場合

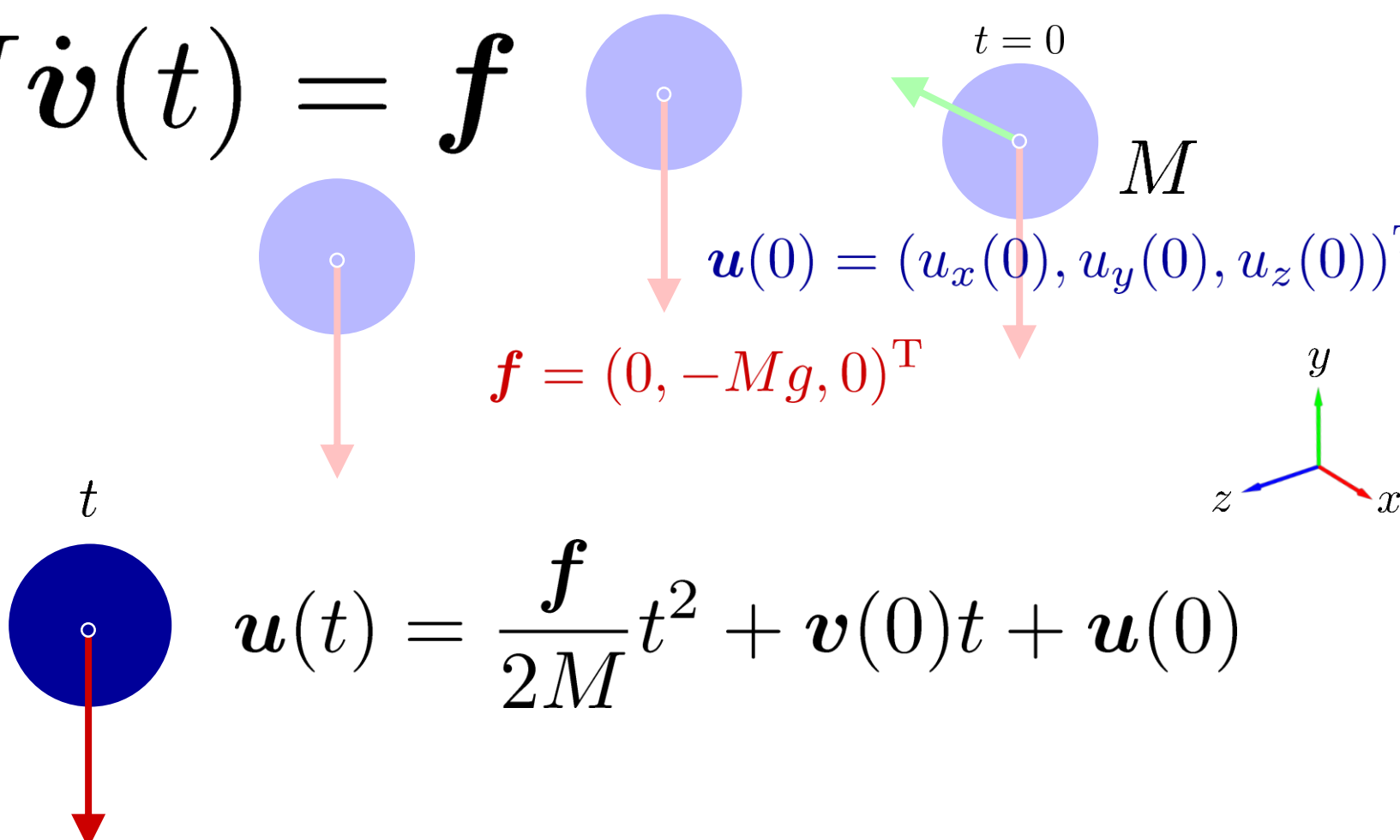
$$M\dot{v}(t) = f$$

$$u(t) = \frac{f}{2M}t^2 + v(0)t + u(0)$$



# 運動方程式：3次元の場合

$$M\dot{\boldsymbol{v}}(t) = \boldsymbol{f}$$



$$\dot{\boldsymbol{v}}(0) = \boldsymbol{v}(0) = (v_x(0), v_y(0), v_z(0))^T$$

$$\boldsymbol{u}(0) = (u_x(0), u_y(0), u_z(0))^T$$

$$\boldsymbol{f} = (0, -Mg, 0)^T$$

$$\boldsymbol{u}(t) = \frac{\boldsymbol{f}}{2M}t^2 + \boldsymbol{v}(0)t + \boldsymbol{u}(0)$$



# 解析的ではなく数值的に解く

$$M\dot{\mathbf{v}}(t) = \mathbf{f} \xrightarrow{\text{離散化}} M \frac{\mathbf{v}[n+1] - \mathbf{v}[n]}{\Delta t} = \mathbf{f}$$

$$\dot{\mathbf{u}}(0) = \mathbf{v}(0) = (v_x(0), v_y(0), v_z(0))^T$$

$$\mathbf{u}(0) = (u_x(0), u_y(0), u_z(0))^T$$

$$\mathbf{v}[n] = \frac{\mathbf{u}[n+1] - \mathbf{u}[n]}{\Delta t}$$

~~$$\mathbf{u}(t) = \frac{\mathbf{f}}{2M}t^2 + \mathbf{v}(0)t + \mathbf{u}(0)$$~~

離散化

$$\mathbf{u}(t) \rightarrow \mathbf{u}[n]$$

$$\dot{\mathbf{u}}(t) \rightarrow \mathbf{v}[n] = \frac{\mathbf{u}[n+1] - \mathbf{u}[n]}{\Delta t}$$

$$\dot{\mathbf{v}}(t) \rightarrow \frac{\mathbf{v}[n+1] - \mathbf{v}[n]}{\Delta t}$$

未知

既知

$$\begin{aligned} \mathbf{v}[n+1] &= \mathbf{v}[n] + \frac{\mathbf{f}}{M} \Delta t \\ \mathbf{u}[n+1] &= \mathbf{u}[n] + \mathbf{v}[n] \Delta t \end{aligned}$$

$$\mathbf{u}[0], \mathbf{v}[0]$$

から始めて繰り返し計算して求める

# C言語プログラムで書いてみる

```
int n = 0;

double M, DT;

Vec3d u, v, fe;

M = 0.1; DT = 0.01;

fe.x = 0; fe.y = -9.8 * M; fe.z = 0;

u.x = 0; u.y = 100; u.z = 0;

v.x = 10; v.y = 10; v.z = 0;

while(1){
    v = v + fe * DT / M;
    u = u + v * DT;
    n++;
}
```

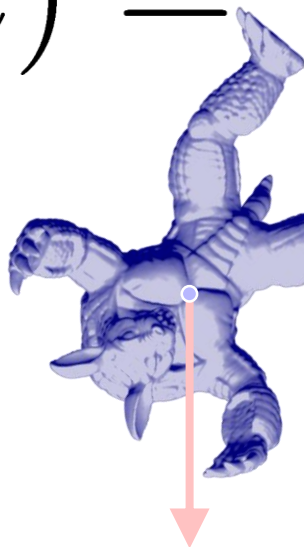
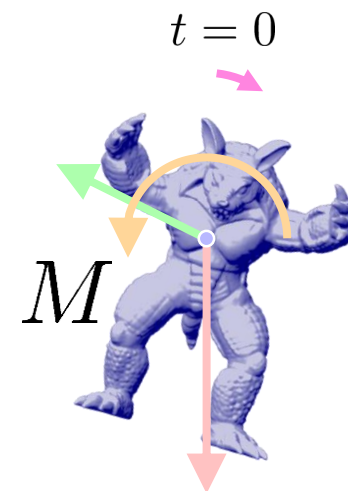
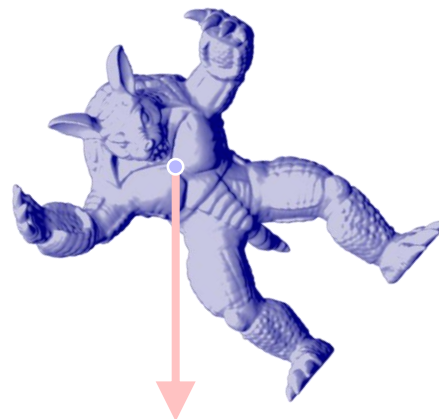
$$\begin{aligned} \boldsymbol{v}[n+1] &= \boldsymbol{v}[n] + \frac{\boldsymbol{f}_e}{M} \Delta t \\ \boldsymbol{u}[n+1] &= \boldsymbol{u}[n] + \boldsymbol{v}[n] \Delta t \\ \boldsymbol{u}[0], \boldsymbol{v}[0] \end{aligned}$$

実際は三次元ベクトルの演算を  
記述しなければならない

# 運動方程式：剛体の場合

$$M\dot{\mathbf{v}}(t) = \mathbf{f}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}}(t) = \boldsymbol{\tau}$$

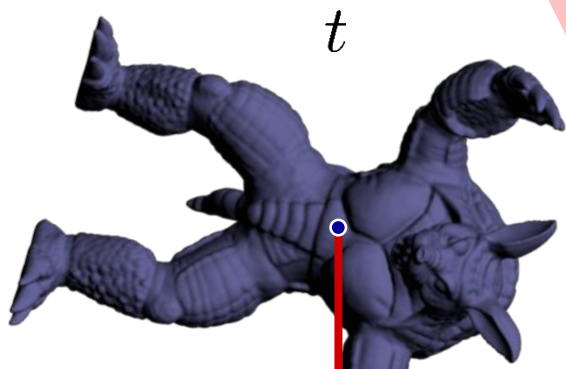


$$\dot{\mathbf{u}}(0) = \mathbf{v}(0) = (v_x(0), v_y(0), v_z(0))^T$$

$$\dot{\boldsymbol{\theta}}(0) = \boldsymbol{\omega}(0) = (\omega_x(0), \omega_y(0), \omega_z(0))^T$$

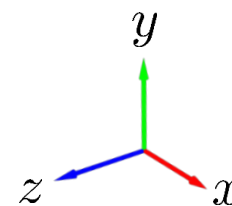
$$\mathbf{u}(0) = (u_x(0), u_y(0), u_z(0))^T$$

$$\boldsymbol{\theta}(0) = (\theta_x(0), \theta_y(0), \theta_z(0))^T$$



$$\mathbf{f} = (0, -Mg, 0)^T$$

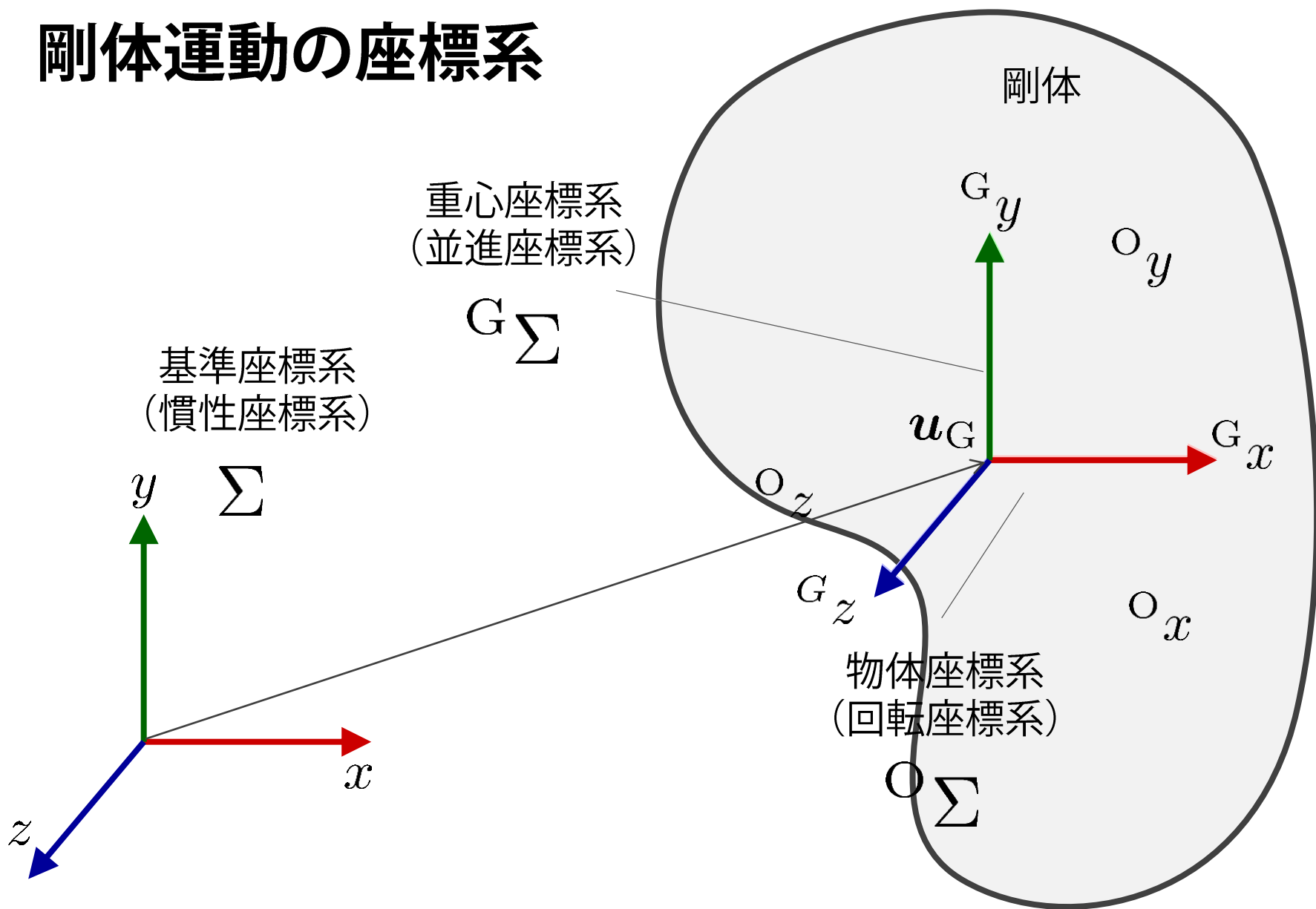
$$\boldsymbol{\tau} = (0, 0, 0)^T$$



# 回転の運動方程式に登場する記号について

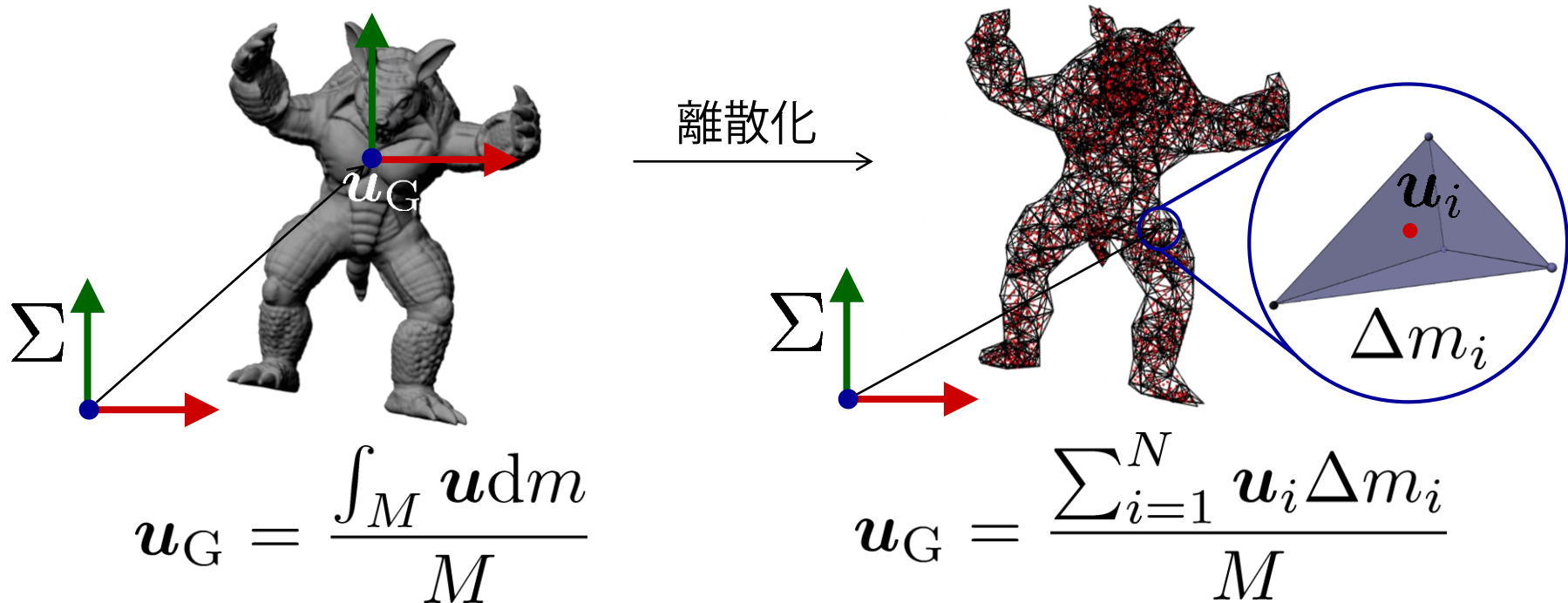
- **$I$**  は何か？
  - 慣性テンソル ( $3 \times 3$ 行列)
  - ある軸の周りで回転させる際の回転のしにくさを表す
- **$\omega$**  は何か？
  - 角速度ベクトル (3次元ベクトル)
  - 回転の方向と回転量を表す
- **$\tau$**  は何か？
  - トルクベクトル (3次元ベクトル)
  - 回転軸まわりの力のモーメント

# 剛体運動の座標系



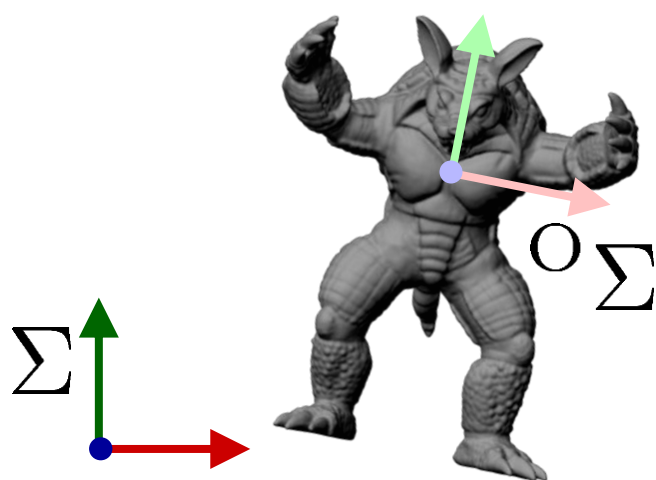
# 質量中心を求める

- 計算機では離散化して扱わなければならない
  - 剛体を四面体要素で表現（ボリウムメッシュ）
  - 四面体中では中心点に質量が集中していると近似

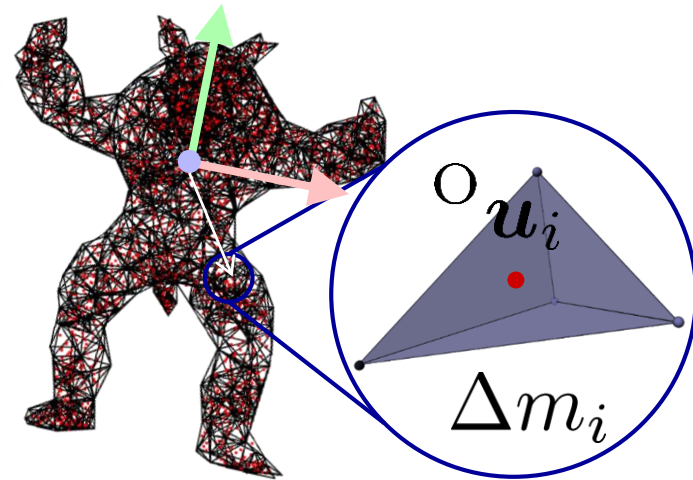




# 物体座標系の慣性テンソル：定数



離散化



$${}^O \mathbf{I} = \int_M {}^O \mathbf{u}^T {}^O \mathbf{u} dm$$

$${}^O \mathbf{I} = \sum_{i=1}^N {}^O \mathbf{u}_i^T {}^O \mathbf{u}_i \Delta m$$

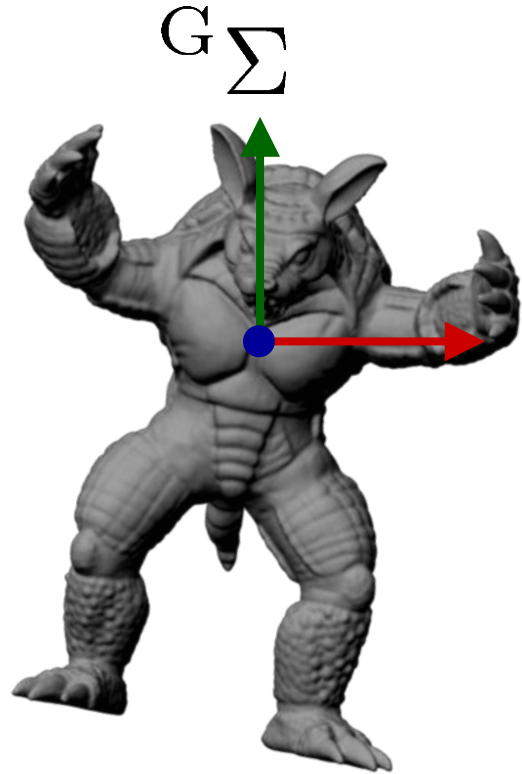
$${}^O \mathbf{u} = \mathbf{R}^{-1}(\mathbf{u} - \mathbf{u}_G)$$

$$= \begin{bmatrix} {}^O I_{xx} & {}^O I_{xy} & {}^O I_{xz} \\ {}^O I_{yx} & {}^O I_{yy} & {}^O I_{yz} \\ {}^O I_{zx} & {}^O I_{zy} & {}^O I_{zz} \end{bmatrix}$$

基準座標系から物体座標系への変換

$$\Sigma \rightarrow {}^O \Sigma$$

# 重心座標系での慣性テンソル：時間変化



$$\mathbf{I} = \mathbf{R}^T \mathbf{O} \mathbf{I} \mathbf{R}$$

添字Gは省略

$${}^G u = \mathbf{R}^O u$$

物体座標系から重心座標系への変換

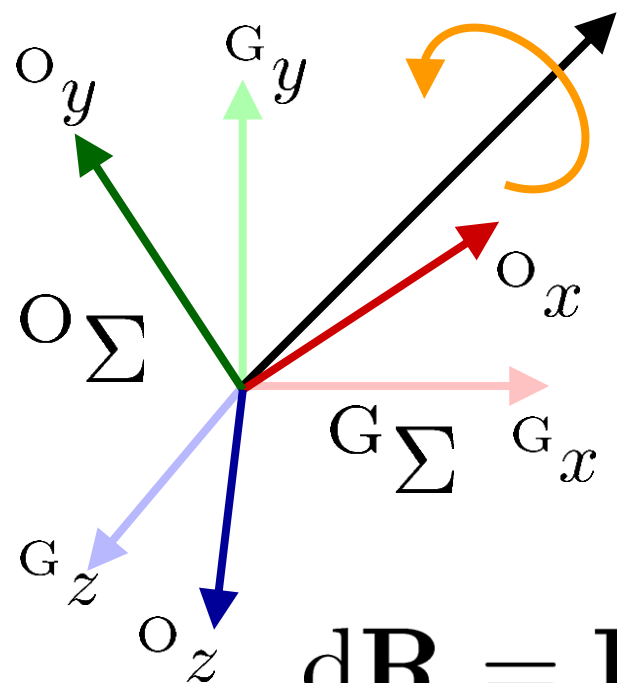
$${}^O \Sigma \rightarrow {}^G \Sigma$$

# 角速度ベクトルと回転行列

添字Gは省略

$$\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$$

$$\omega = |\boldsymbol{\omega}|$$

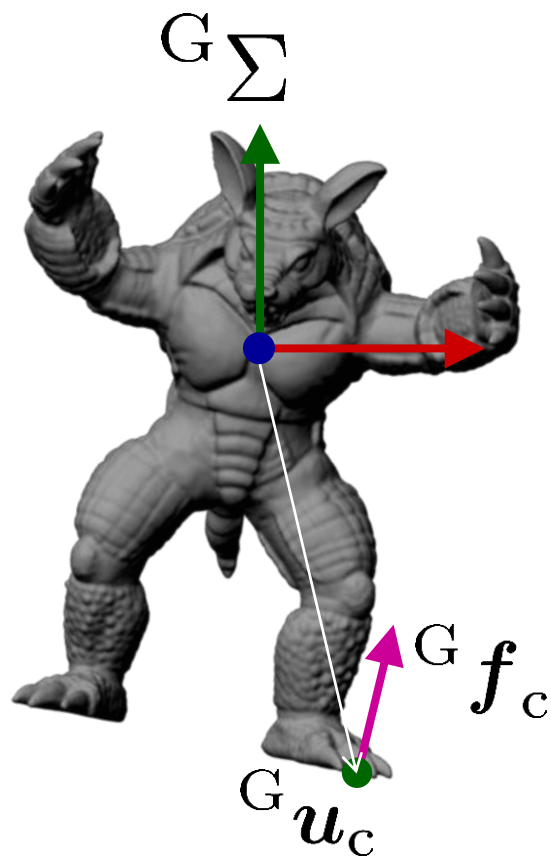


$$\mathbf{W} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

$$d\mathbf{R} = \mathbf{E} + \frac{\sin \omega}{\omega} \mathbf{W} + \frac{1 - \cos \omega}{\omega^2} \mathbf{W}^2$$

$$O_{\Sigma} \rightarrow G_{\Sigma}$$

# 重心座標系におけるトルクベクトル



$$\tau = \sum_c {}^G u_c \times {}^G f_c$$

添字Gは省略

# 並進と回転の運動方程式を連立させて解く

基準・重心座標系でのニュートン・オイラー方程式

$$\begin{bmatrix} M\mathbf{E}_{33} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix}$$

$\mathbf{M}$  6×6 質量行列

$$\mathbf{M}\dot{\mathbf{V}} = \mathbf{F}$$

$\mathbf{V}$  6次元一般化速度ベクトル

1次元の場合と大差はない

$\mathbf{F}$  6次元一般化力ベクトル

# 剛体に働く力

$$\mathbf{F} = \mathbf{F}_E + \mathbf{F}_C$$

- 外力（重力など．拘束力を除く）
  - 問題設定の段階で既知の場合が多い
    - 質量がわかっているれば重力も求まる
- 拘束力（こちらの扱いが難しい）
  - 力の大きさは不明
  - 剛体同士の位置・速度関係が決まっている
    - 抗力：2物体が互いに侵入しない
    - 静止摩擦力：物体が滑らない



# 拘束力をどのように求めるか

- ペナルティ法

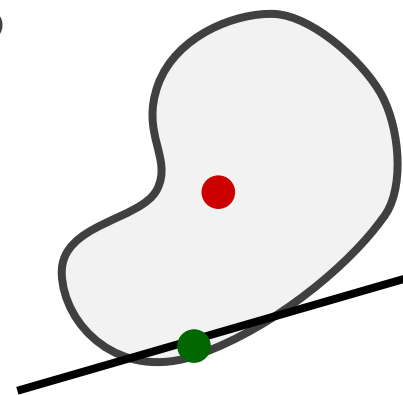
- 侵入量に応じて力を加える
- 適当なばね係数を決める
- 正確さ，安定性は保証されないが，計算は簡単

- 制約法

- 位置や速度を制限し，侵入を防ぐ
- 繰り返し計算により釣り合いの状態を見つける
- 計算が複雑だが比較的安定

# 拘束条件を連立させて解く

- ここまでは重心の並進運動と重心周りの回転運動を考えた
- 接触（拘束）は剛体表面の任意の点で生じ得る
- 拘束点の運動方程式に変換する
- 拘束点に制約条件を与えて拘束力を求める
- 重心の並進運動と重心周りの回転運動を解く



$$M\dot{\mathbf{V}} = \mathbf{F}$$

ニュートン・オイラー運動方程式

$$\mathbf{J}\mathbf{V} = \mathbf{0}$$

接触点の速度がゼロ

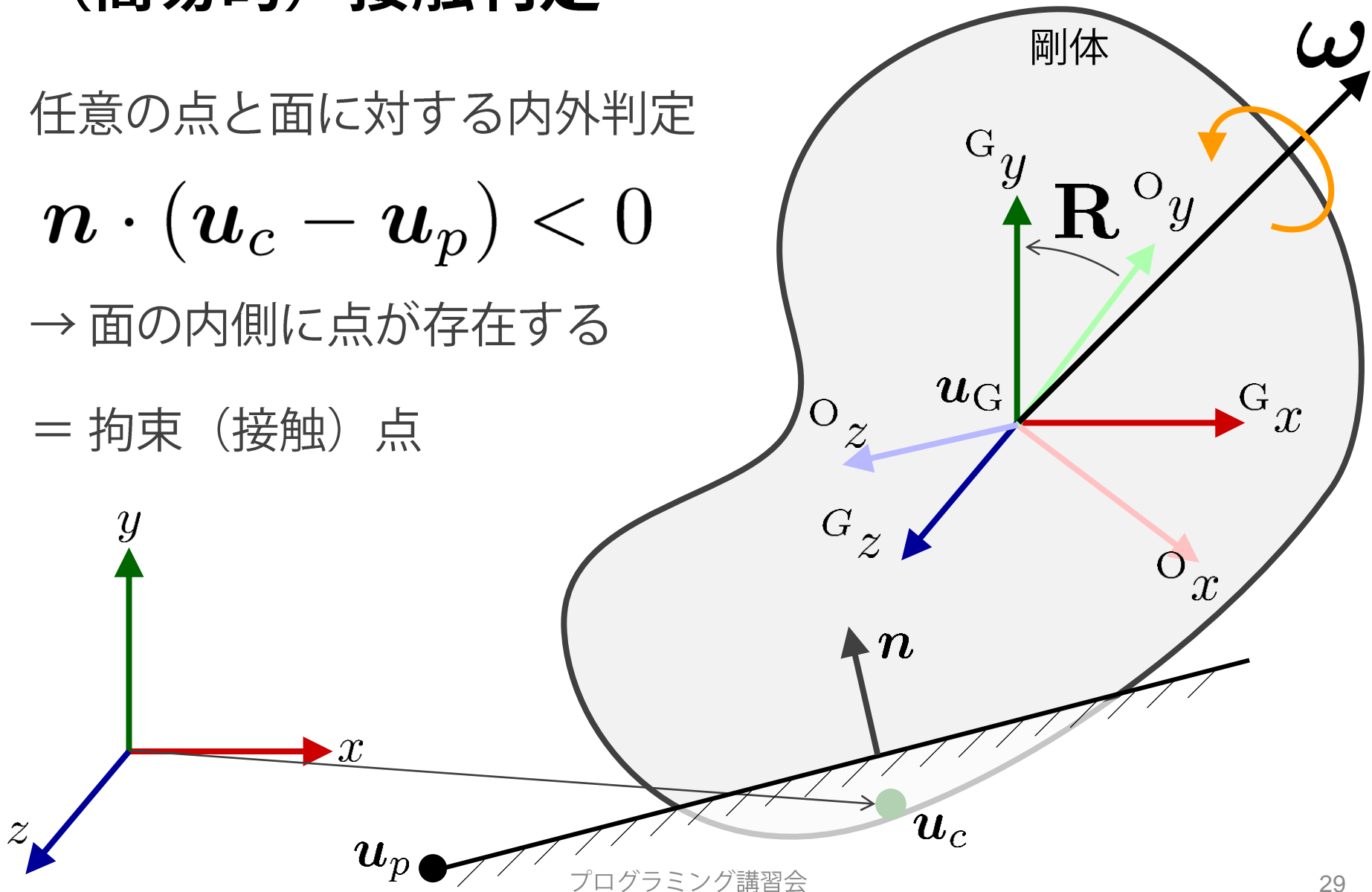
# (簡易的) 接触判定

任意の点と面に対する内外判定

$$\mathbf{n} \cdot (\mathbf{u}_c - \mathbf{u}_p) < 0$$

→ 面の内側に点が存在する

= 拘束 (接触) 点



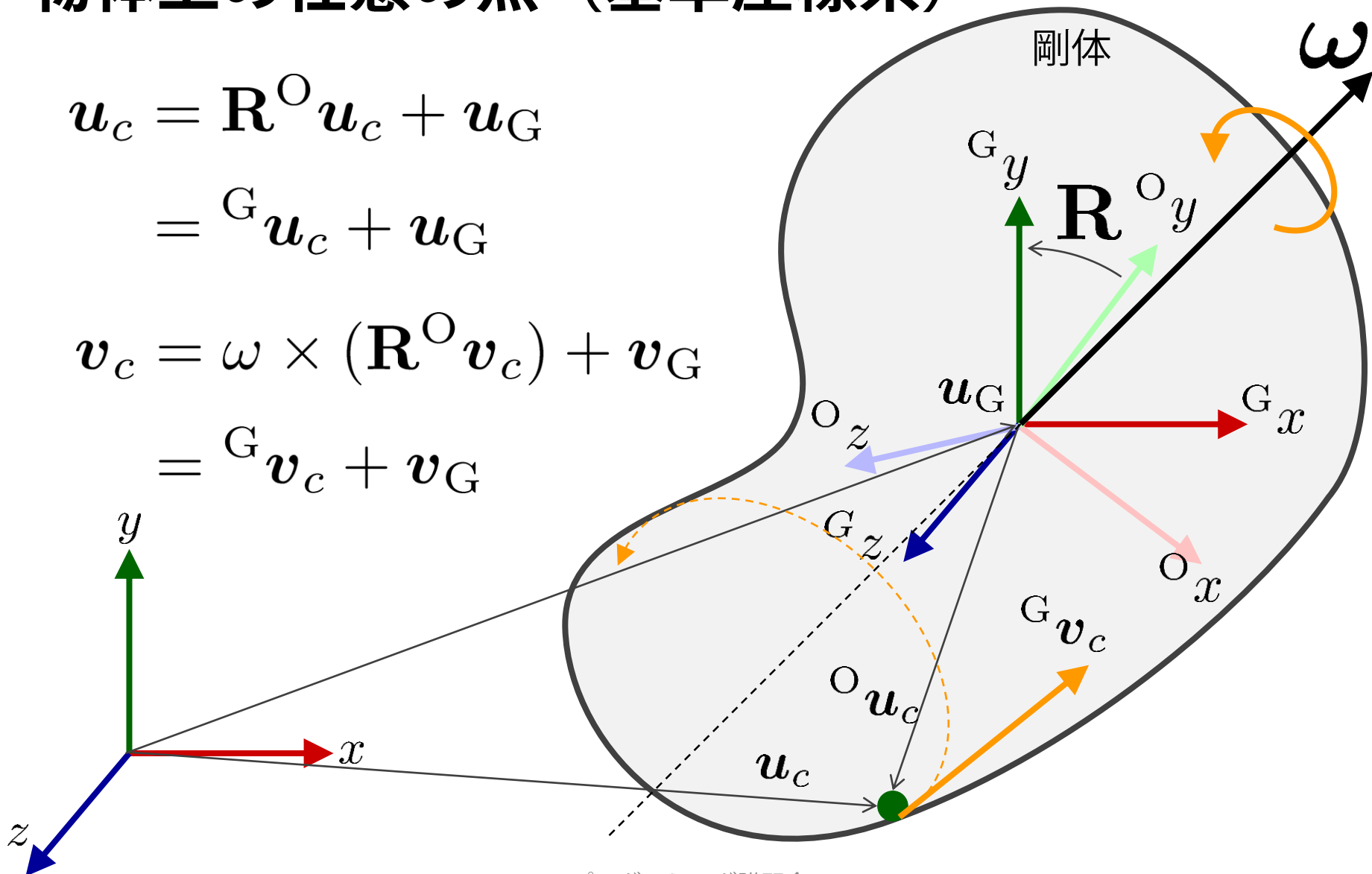
# 物体上の任意の点（基準座標系）

$$\mathbf{u}_c = \mathbf{R}^O \mathbf{u}_c + \mathbf{u}_G$$

$$= {}^G \mathbf{u}_c + \mathbf{u}_G$$

$$\mathbf{v}_c = \boldsymbol{\omega} \times (\mathbf{R}^O \mathbf{v}_c) + \mathbf{v}_G$$

$$= {}^G \mathbf{v}_c + \mathbf{v}_G$$

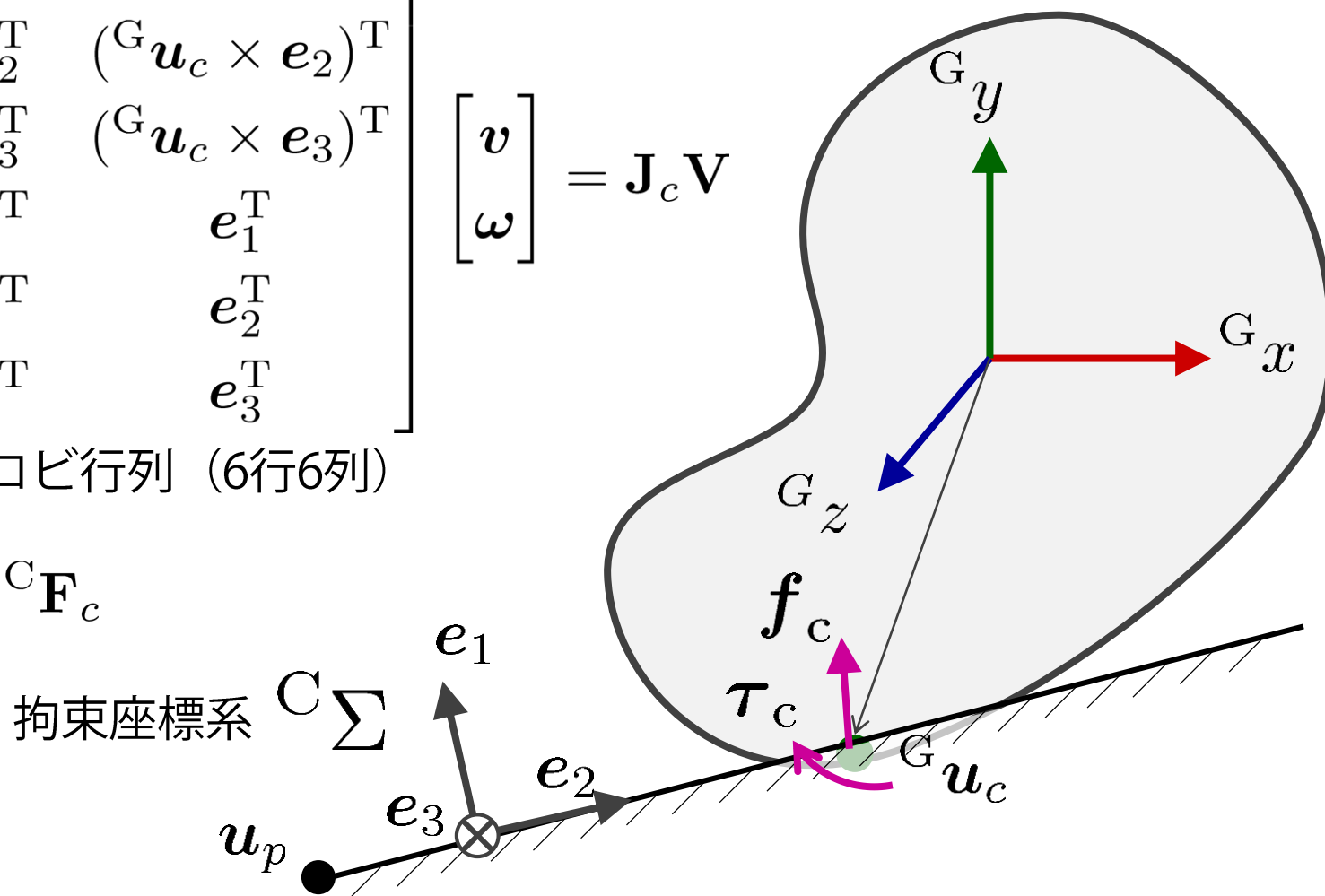


# 拘束点のベクトル表現

$${}^C\mathbf{V}_c = \begin{bmatrix} \mathbf{e}_1^T & ({}^G\mathbf{u}_c \times \mathbf{e}_1)^T \\ \mathbf{e}_2^T & ({}^G\mathbf{u}_c \times \mathbf{e}_2)^T \\ \mathbf{e}_3^T & ({}^G\mathbf{u}_c \times \mathbf{e}_3)^T \\ \mathbf{0}^T & \mathbf{e}_1^T \\ \mathbf{0}^T & \mathbf{e}_2^T \\ \mathbf{0}^T & \mathbf{e}_3^T \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}_c \mathbf{V}$$

ヤコビ行列 (6行6列)

$$\mathbf{F}_c = \mathbf{J}_c^T \mathbf{F}_c$$



# 拘束点における運動方程式

6L行 6列

$$\begin{array}{c}
 \boxed{
 \begin{array}{l}
 {}^C\mathbf{V}_c = \mathbf{J}_c \mathbf{V} \\
 \mathbf{F}_c = \mathbf{J}_c^T {}^C\mathbf{F}_c
 \end{array}
 }
 \xrightarrow[\times L \text{ 点}]{\text{統合}}
 {}^C\mathbf{V}_C = \begin{bmatrix} {}^C\mathbf{V}_1 \\ \vdots \\ {}^C\mathbf{V}_L \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_L \end{bmatrix} \mathbf{V} = \mathbf{J} \mathbf{V} \\
 \mathbf{F}_C = \mathbf{J}^T {}^C\mathbf{F}_C
 \end{array}$$

$$\mathbf{M}\dot{\mathbf{V}} = \mathbf{F}_E + \mathbf{F}_C$$

$${}^C\dot{\mathbf{V}}_C = \mathbf{J}\mathbf{M}^{-1}\mathbf{F}_E + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T {}^C\mathbf{F}_C$$

$${}^C\mathbf{V}_C[n+1] = {}^C\mathbf{V}_C[n] + \mathbf{J}\mathbf{M}^{-1}\Delta t\mathbf{F}_E + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\Delta t {}^C\mathbf{F}_C$$

$${}^C\mathbf{V}_C[n+1] = \mathbf{b} + \mathbf{A} {}^C\mathbf{F}_C$$



# 剛体同士の接触の拘束条件

$${}^C\mathbf{V}_c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ {}^C\omega_{c1} \\ {}^C\omega_{c2} \\ {}^C\omega_{c3} \end{bmatrix} \quad {}^C\mathbf{F}_c = \begin{bmatrix} {}^Cf_{c1} \\ {}^Cf_{c2} \\ {}^Cf_{c3} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- 壁に当たり貫通しない
  - 拘束点に法線力が働く
  - 拘束点の法線速度がゼロになる
  - 壁に引き寄せられる力は働かない
- 剛体間に摩擦力が働く
  - 拘束点に接線力が働く
  - 拘束点の接線速度がゼロになる
  - 滑りが生じている
    - 拘束力は最大静止摩擦力を越えない
    - 接線速度はゼロではない

$$({}^Cf_{c1} > 0) \wedge ({}^C\omega_{c1} = 0)$$

または

$$({}^Cf_{c1} = 0) \wedge ({}^C\omega_{c1} > 0)$$

$$(|{}^Cf_{c2}, {}^Cf_{c3}| < \mu |{}^Cf_{c1}|)$$

$$\wedge ({}^C\omega_{c2}, {}^C\omega_{c3} = 0)$$

または

$$(|{}^Cf_{c2}, {}^Cf_{c3}| = \mu |{}^Cf_{c1}|)$$

$$\wedge ({}^C\omega_{c2}, {}^C\omega_{c3} \neq 0)$$

# ガウスザイデル法で連立方程式を解く

$${}^C\mathbf{V}_C[n+1] = \mathbf{b} + \mathbf{A} {}^C\mathbf{F}_C \rightarrow \mathbf{A} = \mathbf{D} - \mathbf{S}$$

対角成分とそれ以外に分ける

## 1. 初期解を決定する

$$\begin{bmatrix} 0 \\ {}^C\mathbf{V}_{Cl}[n+1] \end{bmatrix} = \begin{bmatrix} \mathbf{b}_f \\ \mathbf{b}_l \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{ff} & \mathbf{A}_{fl} \\ \mathbf{A}_{lf} & \mathbf{A}_{ll} \end{bmatrix} \begin{bmatrix} {}^C\mathbf{F}_{Cf} \\ \mathbf{0} \end{bmatrix}$$

$${}^C\mathbf{F}_{Cf} = -\mathbf{A}_{ff}^{-1} \mathbf{b}_f$$

– 拘束条件を満たすかどうか調べ,  ${}^C\mathbf{F}_C$  を修正する

## 2. 繰り返し計算により解の精度を高める

漸化式を構成する

$${}^C\mathbf{F}_C^{i+1} = \mathbf{D}^{-1} \mathbf{S} {}^C\mathbf{F}_C^i + \mathbf{D}^{-1} ({}^C\mathbf{V}_C[n+1] - \mathbf{b})$$

– 拘束条件を満たすかどうか調べ,  ${}^C\mathbf{F}_C^{i+1}$  を修正する

# 剛体運動のシミュレーションの手順

1. 接触判定
2. 拘束力の計算
3. 剛体の速度・角速度の更新
4. 剛体の位置・姿勢の更新
5. 時刻の更新

# シミュレーション結果

# 変数リスト（剛体パラメータ）

力：	Vec3d force; Vec3d force_constraint; Vec3d force_external;	物体座標系と重心座標系の変換行列：	Matd R, invR;
トルク：	Vec3d torque; Vec3d torque_constraint; Vec3d torque_external;	体積：	double volume;
一般化力ベクトル：	VecNd F; VecNd Fc; VecNd Fe;	質量：	double mass;
位置：	Vec3d position;	質量中心：	Vec3d center_of_mass;
速度：	Vec3d velocity;	慣性テンソル：	Matd I;
角度：	Vec3d angle;	物体座標系での慣性テンソル：	Matd I_o;
角速度：	Vec3d angular_velocity;	質量行列：	Matd M, invM;
一般化速度ベクトル：	VecNd V;	時間ステップ：	double DT;
一般化位置ベクトル：	VecNd U;	時刻：	double time;
		離散時間：	int n;

# 変数リスト（拘束パラメータ）

- 摩擦係数： `double mu;`
- 接触点数： `unsigned int num_contact;`
- 拘束条件かどうかを格納する変数：  
`int *is_constraint;`
- ヤコビ行列： `Matd *Je;`  
`Matd J;`  
`Matd tJ;`
- 拘束座標の一般化速度ベクトル：  
`VecNd V_c;`
- 拘束座標の一般化力ベクトル：  
`VecNd Fc_c;`

# 関数リスト

剛体パラメータの初期化 `void initRigidParams( RigidParams *_rigid );`

剛体パラメータの開放 `void releaseRigidParams( RigidParams *_rigid );`

重心座標系で運動方程式を解く `void solve( RigidParams *_rigid );`

パラメータの読み込み `void loadRigidParams( RigidParams *_rigid, const char *_filename );`

外力の設定 `void setGravity( RigidParams *_rigid, Vec3d _gravity );`

重心座標系の慣性テンソルを算出 `void updateItensor( RigidParams *_rigid );`

重心座標系における質量行列の更新 `void updateMassMatrix( RigidParams *_rigid );`

一般化ベクトルの生成 `void combineVectors( RigidParams *_rigid );`

要素ベクトルへの分解 `void decomposeVectors( RigidParams *_rigid );`

拘束パラメータの初期化 `void initConstraint( Constraint *_constraint );`

拘束パラメータの開放 `void releaseConstraint( Constraint *_constraint );`

接触点の検出 `void collisionTest( Mesh *_mesh, RigidParams *_rigid, Mesh *_container, Constraint *_constraint );`

拘束力の算出 `void solveConstraint( RigidParams *_rigid, Constraint *_constraint );`

# 注意点

- シミュレーションパラメータ
  - 初期条件など，色々と変化させてみましょう
  - 質量はシミュレーション結果に無関係
  - タイムステップは，短いほうが安定
  - 対象物の節点の数が多い（形状が複雑だ）と発散しやすい
- わかり易さを重視して省略した点
  - クォータニオンによる座標回転の表現
  - 物体座標系で回転の運動方程式を記述する
  - 数値積分にルンゲクッタ法などの高精度な方法を採用する
  - 衝突時の撃力の表現，空気抵抗の影響の考慮



物理シミュレーション

# まとめ

# シミュレーションで注意すべきこと

- 解析条件を設計するに至った理由を的確に説明する
  - シミュレーションを行う基礎になるモデルをどのように構築するか
  - 静解析、動解析、周波数解析、座屈解析など、どのようなモードで行うか
  - パラメータや境界条件、入出力は何か
- 現象の性質を示すことはできてもその本質を説明することは難しい
- 妥当性の検証が困難であることを理解する
- 正しく“模擬”しないと、得られた結果は間違いになる
- 現象が複雑だとその間違いに気が付かないこともある

# 参考文献

- 金子成彦・大熊正明 編集, 機械力学ハンドブック, 朝倉書店, 2015年.
- 吉川恒夫 著, ロボット制御基礎論, コロナ社, 2010年.
- 吉川孝雄ら 著, 機械の力学, コロナ社, 2014年.
- Kenny Erleben, Multibody Dynamics Animation Lecture 12, 2005.
- 田浦健次郎, 剛体の物理の基礎,  
[https://www.eidos.ic.i.u-tokyo.ac.jp/~tau/lecture/computational\\_physics/](https://www.eidos.ic.i.u-tokyo.ac.jp/~tau/lecture/computational_physics/)