流体の方程式を援用したCGの演出手法

土橋宜典
北海道大学大学院情報科学研究科
http://ime.ist.hokudai.ac.jp/~doba
doba@ime.ist.hokudai.ac.jp

流体シミュレーション

- リアル
- 計算コストが極めて高い
- どんな結果になるかわからない
- 目的の映像効果を表現することは至難

演出表現のために重要な技術

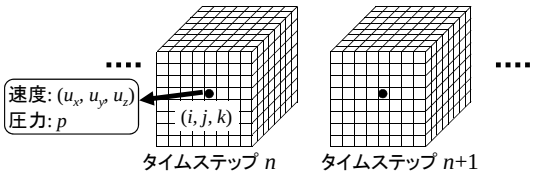
- 高速化
- 流体シミュレーションの制御

高速化

- 速い計算機を使う
- 複数台の計算機を使う（並列計算）
- GPUを使う

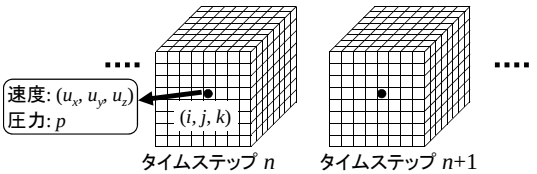
GPUによる高速化

- タイムステップnの状態はタイムステップn-1の状態から決まる
- 格子点ごとに独立に計算可能



GPUによる高速化

- 最新のGPU: 3072演算コア！
- 数十倍～数百倍の高速化が可能

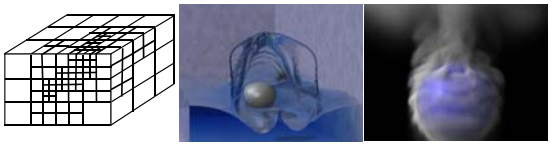


高速化

- アダプティブなシミュレーション
  - 八分木構造の利用
  - 四面体メッシュの利用
  - 重なり格子の利用

アダプティブなシミュレーション

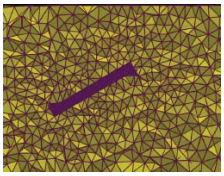
- 八分木構造の利用
  - 複雑な部分を検出して再帰的に分割
  - 複雑でない部分は統合



[Losasso04]

アダプティブなシミュレーション

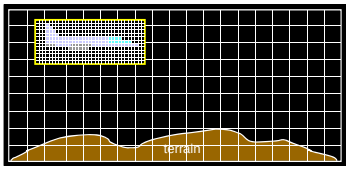
- 四面体メッシュの利用
  - 解析空間を四面体に分割
  - 境界の形状に沿って分割



[Klinger06]

アダプティブなシミュレーション

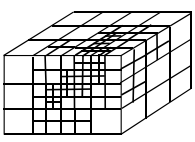
- 重なり格子の利用
  - 重なりあった独立の格子
  - 物体に付随して格子を移動



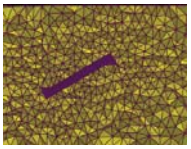
[Dobashi08]

アダプティブなシミュレーション

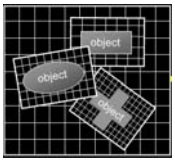
- 精度・計算効率とも向上
- メッシュデータの管理が非常に煩雑
- そこまで速くならない



[Losasso04]



[Klinger06]



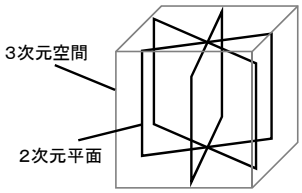
[Dobashi08]

高速化

- 次元を下げる
- 2次元シミュレーションの利用
- データーベースの利用

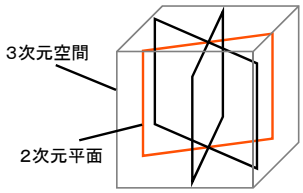
2次元シミュレーションの利用

- 複数の断面を生成



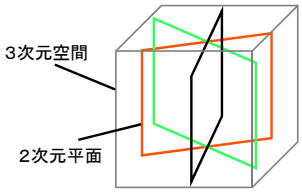
2次元シミュレーションの利用

- 複数の断面を生成
- 2次元シミュレーション



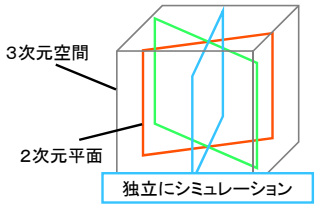
2次元シミュレーションの利用

- 複数の断面を生成
- 2次元シミュレーション



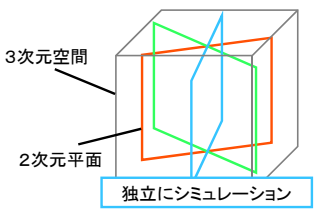
2次元シミュレーションの利用

- 複数の断面を生成
- 2次元シミュレーション



2次元シミュレーションの利用

- 3次元の速度場を補間
- 乱流の特性を考慮



2次元シミュレーションの利用

- 結果

[Rassmussen03]

データベースを利用した高速化

- 低次元化された空間で解く

$$\mathbf{u} = \sum_{i=1}^{m} \mathbf{u}_i r_i = B\mathbf{r} \quad \left(\begin{array}{l} \mathbf{u} \in \mathbf{R}^n \quad \mathbf{r} \in \mathbf{R}^m \\ B\text{: } n \times m\text{の行列} \end{array} \right)$$

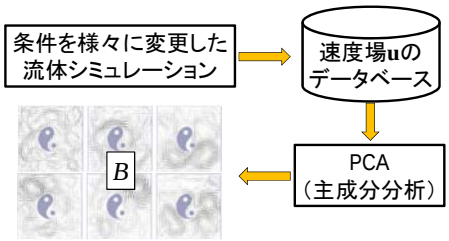
$$\dot{\mathbf{u}} = M\mathbf{u} \iff \dot{\mathbf{r}} = B^T MB\mathbf{r}$$

これを解く

[Treuille04]

データベースを利用した高速化

- 基底速度場行列Bの作成



[Treuille04]

データベースを利用した高速化

- 利点

$$\dot{\mathbf{u}} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \upsilon\nabla^2\mathbf{u} + \mathbf{f}$$

$\nabla\cdot\mathbf{u} = \mathbf{0}$ ← これがやっかい

$\frac{\Delta t}{\rho}\nabla^2 p = \nabla\cdot\tilde{\mathbf{u}}$ ← この計算が遅い

- 基底速度場は $\nabla\cdot\mathbf{u} = \mathbf{0}$ を満足している！

[Treuille04]

データベースを利用した高速化

- 利点

$$\dot{\mathbf{u}} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \upsilon\nabla^2\mathbf{u} + \mathbf{f}$$

$\nabla\cdot\mathbf{u} = \mathbf{0}$ ← これがやっかい

$\frac{\Delta t}{\rho}\nabla^2 p = \nabla\cdot\tilde{\mathbf{u}}$ ← この計算が遅い

- 基底速度場は $\nabla\cdot\mathbf{u} = \mathbf{0}$ を満足している！

[Treuille04]

データベースを利用した高速化

- 利点

$$\dot{\mathbf{u}} = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \upsilon\nabla^2\mathbf{u} + \mathbf{f}$$

$$\dot{\mathbf{u}} = -(\mathbf{u}\cdot\nabla)\mathbf{u} = A\mathbf{u} \iff \dot{\mathbf{r}} = B^T AB\mathbf{r}$$

$$\dot{\mathbf{u}} = \upsilon\nabla^2\mathbf{u} = D\mathbf{u} \iff \dot{\mathbf{r}} = B^T DB\mathbf{r}$$

$$\dot{\mathbf{u}} = \mathbf{f} \iff \dot{\mathbf{r}} = B\mathbf{f}$$

[Treuille04]

データベースを利用した高速化

- 利点

事前に計算可能！

$$\dot{\mathbf{u}} = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \upsilon\nabla^2\mathbf{u} + \mathbf{f}$$

$$\dot{\mathbf{u}} = -(\mathbf{u}\cdot\nabla)\mathbf{u} = A\mathbf{u} \iff \dot{\mathbf{r}} = B^T AB\mathbf{r}$$

$$\dot{\mathbf{u}} = \upsilon\nabla^2\mathbf{u} = D\mathbf{u} \iff \dot{\mathbf{r}} = B^T DB\mathbf{r}$$

$$\dot{\mathbf{u}} = \mathbf{f} \iff \dot{\mathbf{r}} = B\mathbf{f}$$

[Treuille04]

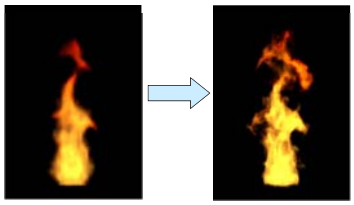
データベースを利用した高速化

- 例

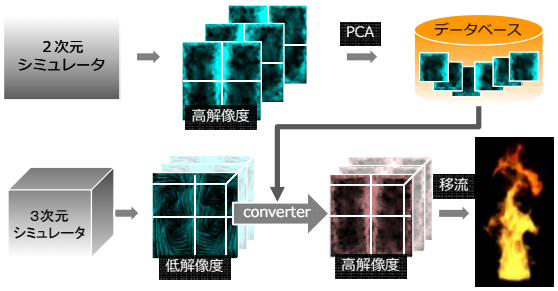
[Treuille04]

データベースを利用した高速化

- 2次元シミュレーションによりデータベースを構築する方法

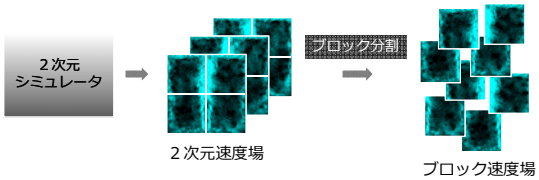


2次元シミュレーションによりデータベースを構築する方法



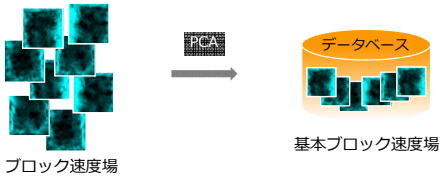
2次元シミュレーションによりデータベースを構築する方法

- データベースの構築



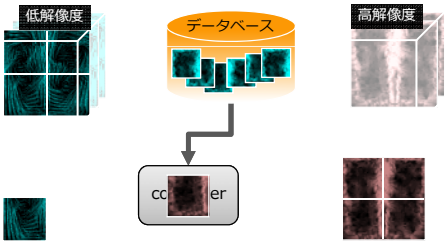
2次元シミュレーションによりデータベースを構築する方法

- データベースの構築

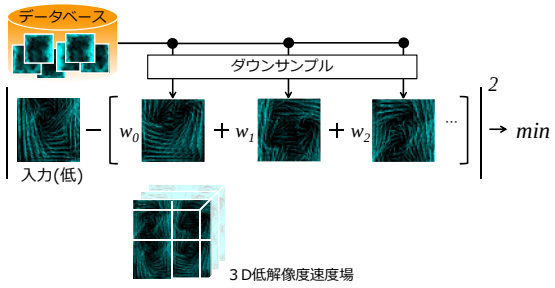


2次元シミュレーションによりデータベースを構築する方法

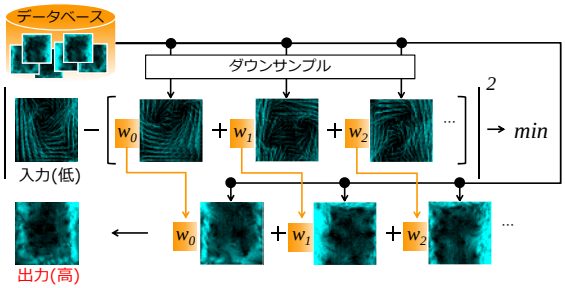
- 高解像度化処理



2次元シミュレーションによりデータベースを構築する方法



2次元シミュレーションによりデータベースを構築する方法



2次元シミュレーションによりデータベースを構築する方法

- 計算時間の比較

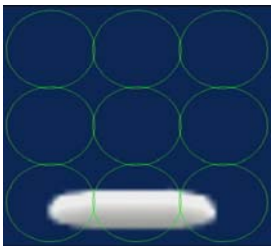
T_{pre}	T_{low}	T_{synth}	T_{high}
48.72 sec	0.05 sec	0.34 sec	0.91 sec

T_{pre} : 前計算　　T_{low} : 低解像度シミュレーション
T_{synth} : 高解像度化処理　　T_{high} : 高解像度シミュレーション

演出表現のために重要な技術

- 高速化
- 流体シミュレーションの制御

外力による煙の制御



[Treuille03]

外力による煙の制御

- 最小化問題として定式化

シミュレーションの1タイムステップ

$\mathbf{q}_t = S(\mathbf{q}_{t-1})$ （$\mathbf{q}_t$: 時刻tの速度場と密度場）

シミュレーションの全体

$\Psi(\mathbf{q}_0) = (\mathbf{q}_1, \mathbf{q}_1, \cdots, \mathbf{q}_n)$

$\Psi(\mathbf{q}_0, \mathbf{u}) = (\mathbf{q}_1, \mathbf{q}_1, \cdots, \mathbf{q}_n)$

（$\mathbf{u}$: 目的の状態を与える外力）

外力による煙の制御

- 最小化問題として定式化

$$\arg\min_{\mathbf{u}} \varphi(\Psi(\mathbf{q}_0,\mathbf{u}),\mathbf{u})$$

$$\varphi(\Psi(\mathbf{q}_0,\mathbf{u}),\mathbf{u}) = \varphi_k(\Psi(\mathbf{q}_0,\mathbf{u})) + \varphi_s(\mathbf{u})$$

φ_k：目標状態との相違を評価する関数

φ_s：加えた力の総和を評価する関数

- 準ニュートン法により解く

外力による煙の制御

- 目的関数と微分

$$\varphi_s = k_s \sum_{t=0}^{n} |\mathbf{f}_t|^2, \quad \frac{d\varphi_s}{du_k} = 2k_s \sum_{t=0}^{n} \mathbf{f}_t \cdot \frac{d\mathbf{f}_t}{du_k}$$

- **f** はガウス関数を用いて表現
- 解析的に微分可能

外力による煙の制御

- 目的関数と微分

$$\varphi_k = k_d \sum_{t\in K_d} |\rho_t - \rho_t^*|^2 + k_v \sum_{t\in K_v} |\mathbf{v}_t - \mathbf{v}_t^*|^2$$

$$\frac{d\varphi_k}{du_k} = 2k_d \sum_{t\in K_d} (\rho_t - \rho_t^*) \frac{d\rho_t}{du_k} + 2k_v \sum_{t\in K_v} (\mathbf{v}_t - \mathbf{v}_t^*) \frac{d\mathbf{v}_t}{du_k}$$

外力による煙の制御

- 目的関数と微分

$$\varphi_k = k_d \sum_{t\in K_d} |\rho_t - \rho_t^*|^2 + k_v \sum_{t\in K_v} |\mathbf{v}_t - \mathbf{v}_t^*|^2$$

$$\frac{d\varphi_k}{du_k} = 2k_d \sum_{t\in K_d} (\rho_t - \rho_t^*) \frac{d\rho_t}{du_k} + 2k_v \sum_{t\in K_v} (\mathbf{v}_t - \mathbf{v}_t^*) \frac{d\mathbf{v}_t}{du_k}$$

シミュレーションにより得られる密度と速度の微分

外力による煙の制御

- シミュレーションの微分

$$\mathbf{q}_t = S(\mathbf{q}_{t-1})$$

$$S = M \circ A_\rho \circ P \circ D \circ A_\mathbf{v} \circ F$$

数値拡散の補正　移流（密度）　非圧縮性　拡散　移流（速度）　外力

- 各項について**q**の微分を計算しながら更新

外力による煙の制御

- 非圧縮性（P）と拡散（D）
  - いずれも線形な演算子

$$\frac{dP}{du_k}(\mathbf{v}) = P\left(\frac{d\mathbf{v}}{du_k}\right)$$

$$\frac{dD}{du_k}(\mathbf{v}) = D\left(\frac{d\mathbf{v}}{du_k}\right)$$

外力による煙の制御

- 移流(A)

$$A(\mathbf{v},\sigma,\mathbf{p}_0) = I(\sigma,\mathbf{p}_s), \quad \mathbf{p}_i = \mathbf{p}_{i-1} - \Delta t I(\mathbf{v},\mathbf{p}_{i-1})$$

$$\frac{dA}{du_k}(\mathbf{v},\sigma,\mathbf{p}_0) = \frac{dI}{du_k}(\sigma,\mathbf{p}_s)$$

$$\frac{d\mathbf{p}_i}{du_k} = \frac{d\mathbf{p}_{i-1}}{du_k} - \Delta t \frac{dI}{du_k}(\mathbf{v},\mathbf{p}_{i-1})$$

外力による煙の制御

- 数値拡散の補正(M)

$$\rho' = \rho \frac{m}{\sum \rho}$$

$$\frac{d\rho'}{du_k} = \frac{d\rho}{du_k}\frac{m}{\sum\rho} - \rho\frac{m}{\left(\sum\rho\right)^2}\sum\frac{d\rho}{du_k}$$

外力による煙の制御

- 外力(F)

$$F(\mathbf{v}) = \mathbf{v} + \mathbf{f}$$

$$\frac{dF}{du_k}(\mathbf{v}) = \frac{d\mathbf{v}}{du_k} + \frac{d\mathbf{f}}{du_k}$$

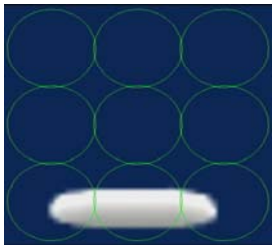
外力による煙の制御

```
EVALUATE(u)
  (φ, dφ/du) ← (0, 0)
  (ρ, v, dρ/du, dv/du) ← (ρ0, v0, 0, 0)

  for t ← 1 to n
    – derivative calculation
    for each active control uk
      df/duk ← FORCEDERIV(t, dv/duk, u)
      dv/duk ← dv/duk + df/duk
      dv/duk ← ADVECTDERIV(v, v, dv/duk)
      dv/duk ← DIFFUSE(dv/duk)
      dv/duk ← PROJECT(dv/duk)
      dρ/duk ← ADVECTDERIV(v, ρ, dρ/duk)
      dρ/duk ← PRESERVEMASSDERIV(ρ, dρ/duk)
```

外力による煙の制御

```
    – fluid simulation
    f ← FORCE(t, u)
    v ← v + f
    v ← ADVECT(v, v)
    v ← DIFFUSE(v)
    v ← PROJECT(v)
    ρ ← ADVECT(v, ρ)
    ρ ← PRESERVEMASS(ρ)

    – objective function
    φ ← φ + OBJECTIVE(t, ρ, v, f)
    for each active control uk
      dφ/duk ← dφ/duk
              + GRAD(t, ρ, dρ/duk, v, dv/duk, f, df/duk)
  return (φ, dφ/du)
```

外力による煙の制御



[Treuille03]

外力による煙の制御

- 目的形状に導く外力を自動計算
- 最適化に数時間必要（2～24時間）
- 制御されている感が拭えない（不自然）

ポテンシャル場による制御

- 目的形状からポテンシャル場を計算
  - 目的形状のフィルタリング処理
- ポテンシャル場の勾配を外力として付加

[Fattal04]

ポテンシャル場による制御

- 与えられるデータ
  - 初期の煙の密度分布 ρ_0
  - 目的の煙の密度分布 ρ^*

ρ_0 ρ^*

ポテンシャル場による制御

- ポテンシャル場を使う利点

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \upsilon\nabla^2\mathbf{u} + \mathbf{f} \quad \nabla\cdot\mathbf{u} = \mathbf{0}$$

ポテンシャル場による制御

- ポテンシャル場を使う利点

$$\mathbf{u} = \mathbf{0} \quad \frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \upsilon\nabla^2\mathbf{u} + \mathbf{f} \quad \nabla\cdot\mathbf{u} = \mathbf{0}$$

$$\mathbf{f} = \nabla p$$

ポテンシャル場による制御

- ポテンシャル場を使う利点

$$\mathbf{u} = \mathbf{0} \quad \frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \upsilon\nabla^2\mathbf{u} + \mathbf{f} \quad \nabla\cdot\mathbf{u} = \mathbf{0}$$

$$\mathbf{f} = \nabla p$$

- $\mathbf{f}$ をポテンシャルの勾配にしておけば，目的形状に達し，$\mathbf{u}$=0となって安定する

ポテンシャル場による制御

- ポテンシャル場の設計

  目的形状の方向に力を発生

$$F(\rho,\rho^*) \propto \nabla\rho^*$$

  0となる場所がなく，弱過ぎない

$$F(\rho,\rho^*) \propto \frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*} \quad (\tilde{\rho}^* = G * \rho^*)$$

ポテンシャル場による制御

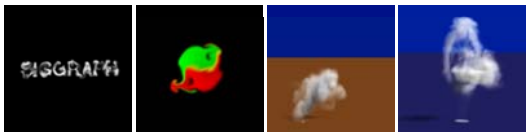
- ポテンシャル場の設計

  煙のないところに力を加えない

$$F(\rho,\rho^*) = \rho\frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*}$$

ポテンシャル場による制御

- 例



外力によらない制御

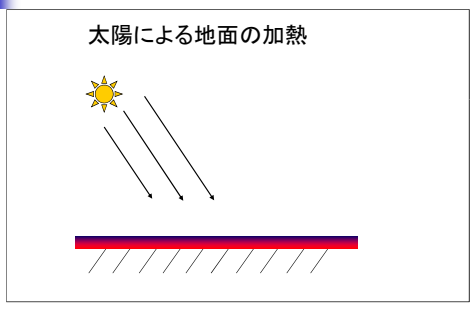
- 雲のシミュレーションの制御

[Dobashi04]

外力によらない制御

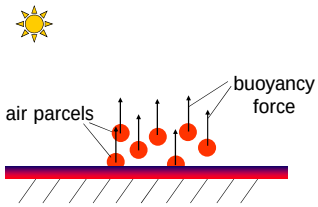
- 雲のシミュレーションの制御

応用3：雲の形成



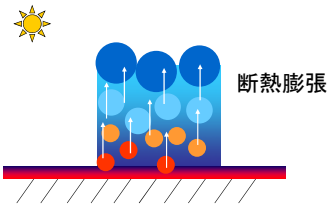
応用3:雲の形成

空気塊の上昇



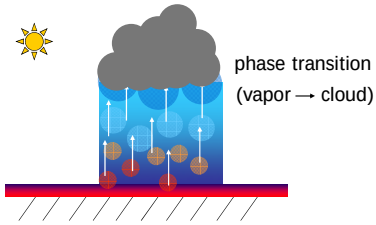
応用3:雲の形成

上昇にともなう温度の減少

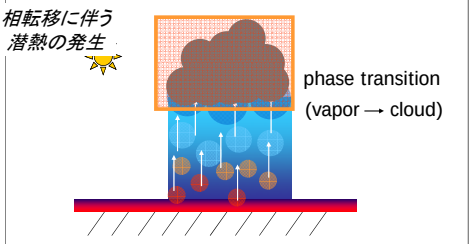


応用3:雲の形成

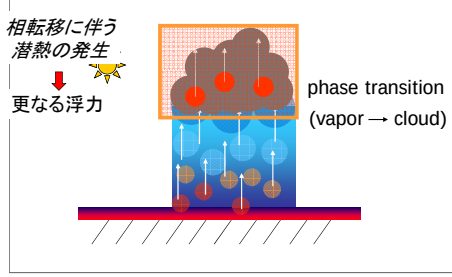
相転移による雲の発生



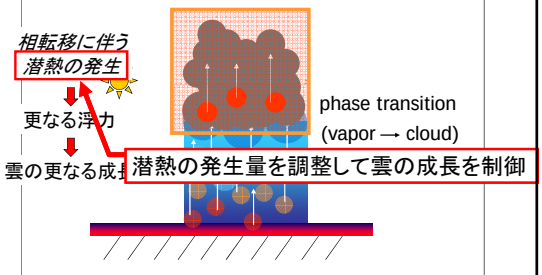
応用3:雲の形成



応用3:雲の形成

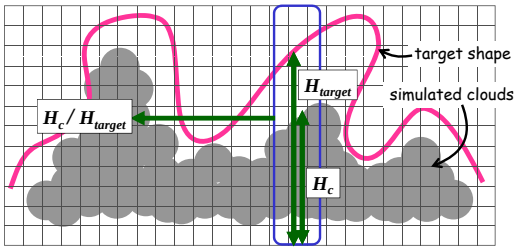


応用3:雲の形成



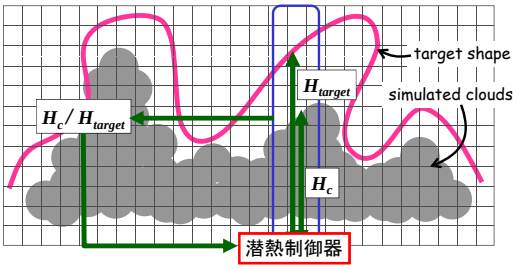
フィードバック制御

- 潜熱制御器と水蒸気供給器



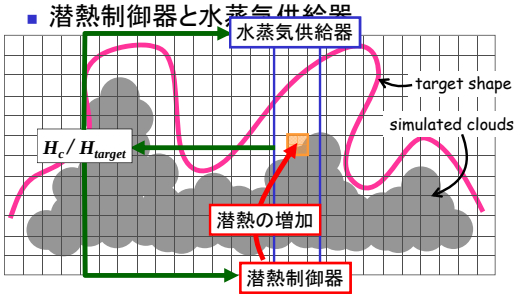
フィードバック制御

- 潜熱制御器と水蒸気供給器



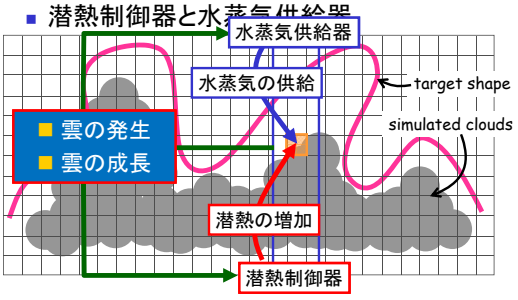
フィードバック制御

- 潜熱制御器と水蒸気供給器



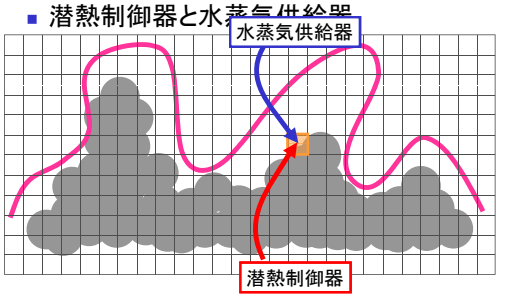
フィードバック制御

- 潜熱制御器と水蒸気供給器



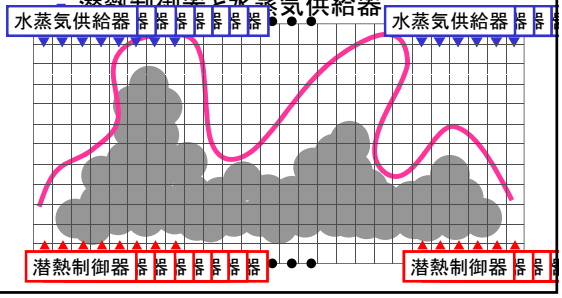
フィードバック制御

- 潜熱制御器と水蒸気供給器

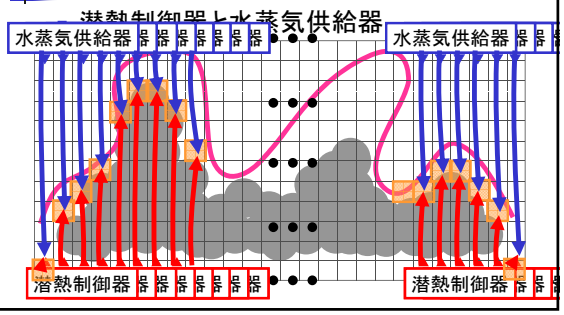


フィードバック制御

- 潜熱制御器と水蒸気供給器



フィードバック制御



大気の速度

- 非圧縮性ナビエ・ストークス方程式

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \mathbf{f} + \mathbf{B}$$

(浮力)

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

$\mathbf{u}$: velocity　$\mathbf{f}$: external force　p : pressure　t : time

浮力と温度

- 熱浮力

$$\mathbf{B} = k_b \frac{T - T_0}{T_0}\mathbf{z}$$

T : 温度　T_0 : 環境温度

$\mathbf{z}$: 鉛直上向きベクトル　k_b : 浮力係数

- 浮力の変化

$$\frac{\partial T}{\partial t} = -(\mathbf{u}\cdot\nabla)T - \Gamma_d v_z + Q C_c + S_T$$

潜熱制御器

Q : 潜熱係数　C_C : 雲の発生量

Γ_d : 乾燥断熱減率　v_z : 速度の鉛直成分

S_T : 地面からの熱

水蒸気と水滴

- 相転移

水滴: $\frac{\partial q_c}{\partial t} = -(\mathbf{u}\cdot\nabla)q_c + C_c$　$C_c = \alpha(q_v - q_s)$

水蒸気: $\frac{\partial q_v}{\partial t} = -(\mathbf{u}\cdot\nabla)q_v - C_c + S_v$

水蒸気供給器

q_c : 雲密度　q_v : 水蒸気密度

C_c : 相転移によって発生する雲の量

潜熱制御器

- PI制御器(P:比例, I:積分)
  - P制御:差に比例して調整
  - I制御:差の積分値に比例して調整

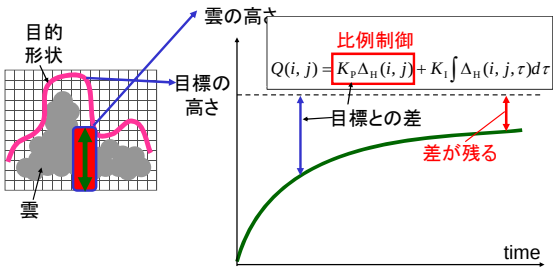
潜熱係数　目的形状との差

$$Q(i,j) = K_P \Delta_H(i,j) + K_I \int \Delta_H(i,j,\tau)d\tau$$

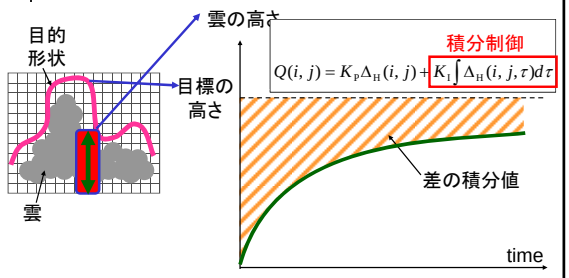
比例制御　積分制御

(K_P:比例ゲイン, K_I: 積分ゲイン)

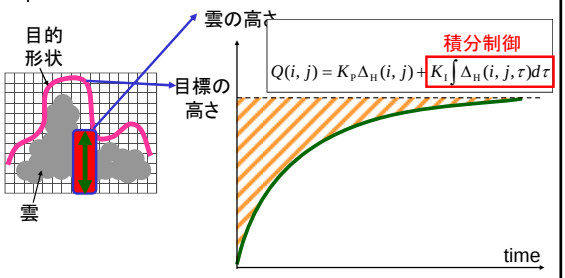
潜熱制御器



潜熱制御器

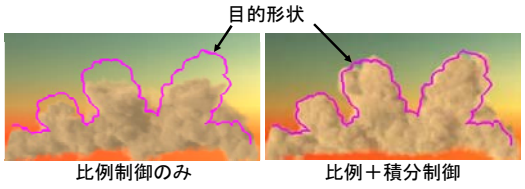


潜熱制御器



潜熱制御器

- 比較



水蒸気供給器

- 水蒸気供給条件

$$\frac{1}{N}\sum(R(i,j))>R(i,j)$$

割合の平均

$$R(i,j)=H_c(i,j)/H_{target}(i,j)$$

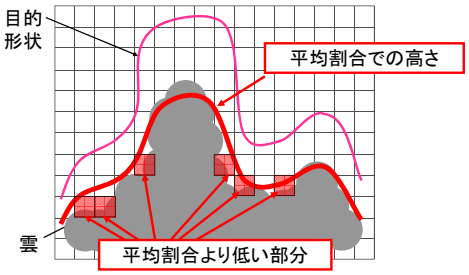
雲の高さ　目標の高さ

- 水蒸気供給量

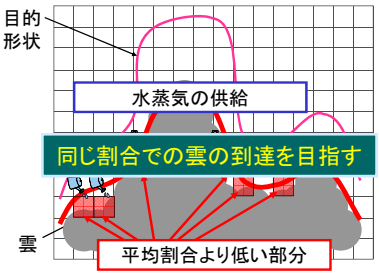
$$S_{v,c}(i,j)=c_v q_{v,0}(i,j,k_{top})$$

ユーザー指定　初期状態での水蒸気量

水蒸気供給器



水蒸気供給器



雲のシミュレーションの制御

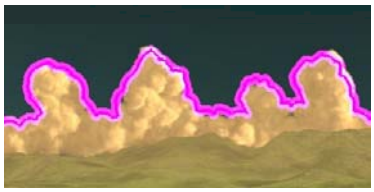
- 比較

ポテンシャル場による制御

物理パラメータの制御

雲のシミュレーションの制御

- 例1



雲のシミュレーションの制御

- 例2

雲のシミュレーションの制御

- 例3

雲のシミュレーションの制御

- 失敗例

流体シミュレーションの制御

- 与えられた目的を自動的に形成
- しかし，試行錯誤はなくならない
  - 制御パラメータの存在
- 低解像度シミュレーションで試行錯誤
- 高解像度シミュレーションが低解像度シミュレーションと類似するよう制御

低解像度による高解像度の制御

[Nielsen09]

まとめ

- しっかりした手法を考案する上で数学は大変重要
- CGで使われている数学は比較的基礎的なもの（だと思います）
- もっといろいろ教えてください