

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

研究進捗

武藤 克弥 (Katsuya Mutoh)
u255018@st.pu-toyama.ac.jp

富山県立大学 大学院 電子・情報工学専攻 情報基盤工学部門

December 22, 2023

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

流行分析の例 1

遅延微分方程式とベイズ推定を用いた流行分析がある

流行分析の例 2

- 「キーワードが“注目”という資源を奪い合う」という生物的なモデル¹を用いると、複数キーワードの流行り廃りをよく説明できる。

目的

【仮定】キーワードは生き残ろうと価値関数を最大にする戦略をとっている人間の戦略的意意思決定モデルをキーワードの流行り廃りに当てはめ、流行分析モデルを作成する

¹Lorenz-Spreen, P., Mønsted, B.M. et al., 2019

ロトカ・ウォルテラ的 注目モデル

「キーワードが注目」という資源を奪い合う」という仮定のもと、作成されたモデル

- 2013年～2016年、ハッシュタグ使用回数上位50単語の1日ごとの推移をモデルにフィッティング

ロトカ・ウォルテラ-注目モデル

$$\frac{dL_i(t)}{dt} = r_p L_i(t) \left(1 - \frac{r_c}{K} Y_i(t) - c \sum_{j=1, j \neq i}^N L_j(t) \right)$$

$$\frac{dY_i(t)}{dt} = L_i(t) - \alpha Y_i(t).$$

$L_i(t)$: ハッシュタグ i の出現回数

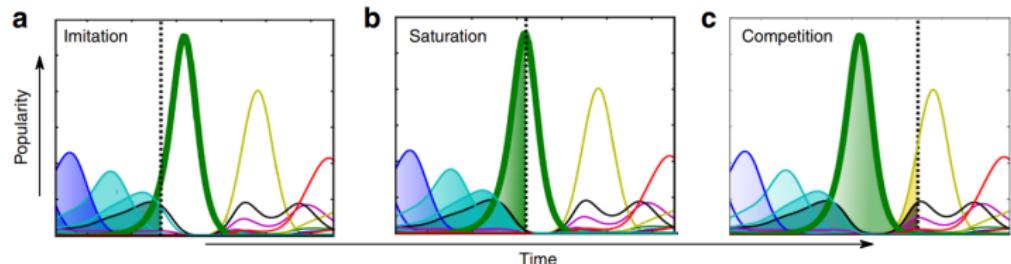
Y : 補助変数

N トピック数

K : 成長モデルの収容力

c : 重複コンテキストの量

r_p, r_c : 1 トピック当たりの成長率



確率分布のフィッティング

キーワードの増減について、実データ分布と各種確率分布と比較
→対数正規分布が最も優れたフィット

研究背景

既存研究

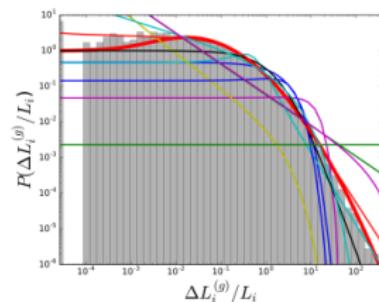
提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

Twitter



blue	expon	blue	logistic
green	powerlaw	green	uniform
red	lognorm	red	fisk
cyan	cauchy	cyan	weibull_min
magenta	norm	magenta	truncated powerlaw
yellow	gamma		
grey	pareto		

その他データセット でも同様

Source	Sampling
Twitter	Top 50 of each hour, sorted by hourly volume
Books	Top 1000 of each year, sorted by relative yearly volume
Movies	Popular movies of each week, sorted by box-office sales
Google	Top 20 of each month, sorted by total queries
Reddit	Top 1000 of each month, sorted by accumulated comments
Publications	More than 15 citations, once in the observation window
Wikipedia	Top 100 of every hour, sorted by traffic per article

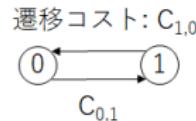
提案手法

5/31

意思決定モデルの応用

市場参入するか撤退するかのモデルを応用

研究背景
既存研究
提案手法
離散化
アルゴリズムの再現
偏微分方程式の差分化



- ①様子見
=他キーワードの注目を探る
≒廃れ状態?
②注目されるための活動
→注目維持のためのコスト w_1

利潤(人の注目度) $\pi_1 = xP - w_1$
(x:定数)

P : 人の注目変動

t : 時間

α : ドリフト係数

σ : 拡散係数

$$dP(t)/P(t) = \alpha dt + \sigma dZ(t)$$

選択肢

- コスト $C_{0,1}$ を支払って ① → ②
- コスト $C_{1,0}$ を支払って 撤退 ② → ①
- ①を維持
- ②を維持

→時刻・状態(t, P)で ①②どちらの戦略をとるべきか
最適値関数 $V0, V1$ で評価

意思決定モデルの応用

市場参入するか撤退するかのモデルを応用

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

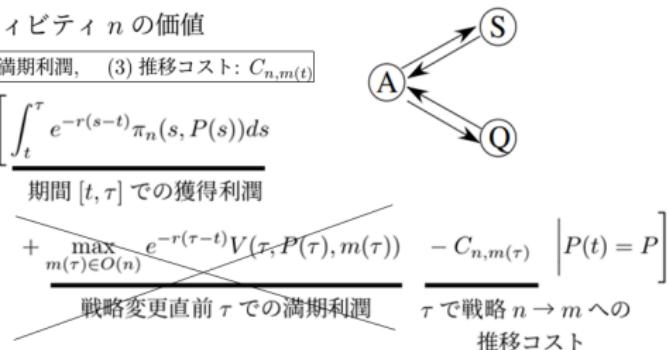
用いる意思決定モデル

$V(t, P, n)$: 時刻 t におけるアクティビティ n の価値

→ 3要素で構成 (1) 利潤: $\pi_n(t, P)$, (2) 満期利潤, (3) 推移コスト: $C_{n,m(t)}$

$$V(t, P, n) = \max_{\tau \in [t, T]} E \left[\int_t^{\tau} e^{-r(s-t)} \pi_n(s, P(s)) ds \right]$$

- r : 割引率
- $n \in \{A, S, Q\}$: アクティビティ
- $O(n)$: アクティビティ n の推移先集合



現在の状況

7/31

著者への質問

著者の方と直接お話し、意思決定アルゴリズムの問題解決に取り組んだ

- アルゴリズムの自己解釈が数か所間違っており、ご指導いただいた
→アルゴリズム修正中

[Algo-Peng-Lin]

Step 0. Given constant numbers $\varepsilon_0 \geq 0$, $\omega \in (0, 1)$, $\delta_1 \in (0, 1)$, $\delta_2 \in (0, 1)$, $\delta_3 \in (0, 1 - \delta_2)$.

Choose any $\xi^{(0)} > 0$, $V^{(0)} \in \mathcal{R}^J$ and $\gamma \geq \frac{\|H(V^{(0)}, \xi^{(0)})\|}{\min\{\xi^{(0)}, 1\}}$.

Step 1. The Newton step of $H(V, \xi^{(k)})$:

If $\nabla_V H(V^{(k)}, \xi^{(k)})$ is singular, (the algorithm fails);

else if $\|H(V^{(k)})\| \leq \varepsilon_0$, ($V^{(k)}$ is an appropriate solution of [GLCP]);

Otherwise, compute a Newton step $d^{(k)}$ satisfying Eq. (37).

Step 2. Compute $V^{(k+1)}$:

Let $h^{(k)}$ be the maximum value of $\{1, \delta_1, \delta_1^2, \dots\}$ such that

$$\|H(V^{(k)} + h^{(k)} d^{(k)}, \xi^{(k)})\| \leq (1 - \omega h^{(k)}) \min\{\xi^{(k)}, 1\} \gamma, \quad (\text{B.1})$$

and $V^{(k+1)} := V^{(k)} + h^{(k)} d^{(k)}$.

Step 3. Compute $\xi^{(k+1)}$:

If $(V^{(k+1)}, \min\{\delta_3, \xi^{(k)}\} \xi^{(k)}) \in \mathcal{N}(\gamma, \min\{\delta_3, \xi^{(k)}\} \xi^{(k)})$ then set $v^{(k)} := 1 - \min\{\delta_3, \xi^{(k)}\}$;

Otherwise, let $v^{(k)}$ be the maximum value of $\{\delta_2, \delta_2^2, \dots\}$ such that

$$(V^{(k+1)}, (1 - v^{(k)}) \xi^{(k)}) \in \mathcal{N}(\gamma, (1 - v^{(k)}) \xi^{(k)}). \quad (\text{B.2})$$

Set $\xi^{(k+1)} := (1 - v^{(k)}) \xi^{(k)}$.

Step 4. $k := k + 1$, return to Step 1.

where in the above

$$\mathcal{N}(\beta, \xi) = \{(V, \xi) \in \mathcal{R}^J \times \mathcal{R}_+ \mid \|H(V, \xi)\| \leq \beta \min\{\xi, 1\}\}. \quad (\text{B.3})$$

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

提案手法(破棄)

8/31

やりたいこと

市場参入するか撤退するかのモデルを応用

研究背景

既存研究

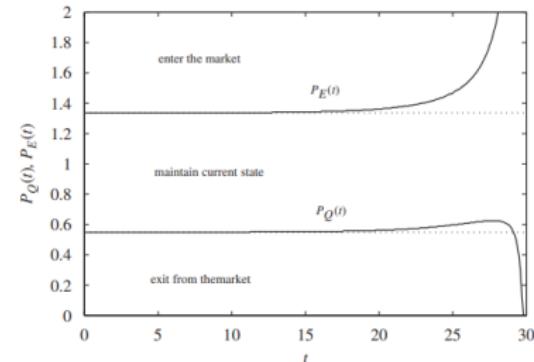
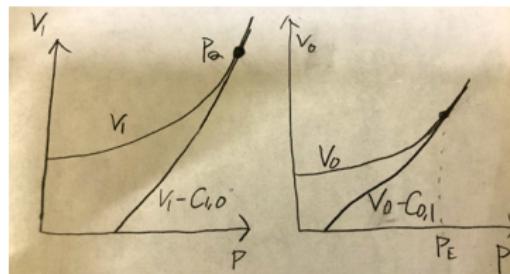
提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

1単語の戦略変動



各キーワードa, b, cの時刻・状態(t,P)の最適値Va, Vb, Vcが分かる

Vが最高値のキーワードが生き残る一ロトカ・ヴォルテラのようなモデルで評価

単語Va

遷移コスト: $C_{1,0}$



$C_{0,1}$

単語Vb

遷移コスト: $C_{1,0}$



$C_{0,1}$

単語Vc

遷移コスト: $C_{1,0}$



$C_{0,1}$

提案手法

9/31

やりたいこと

ある単語がいつ状態 1 と状態 2 を切り替えていたのか分析

研究背景

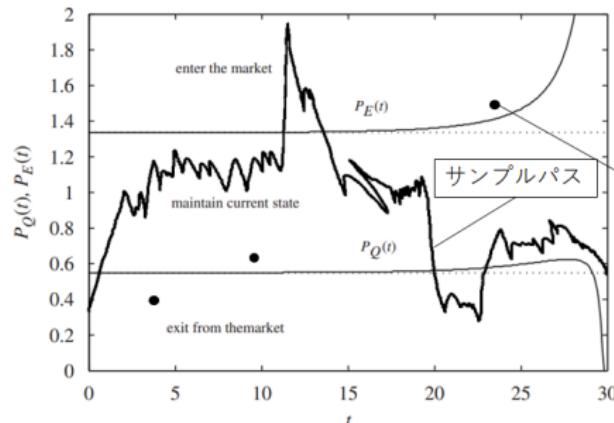
既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化



金融の場合

サンプルパスは大体分かる ↓

$$dP(t)/P(t) = \alpha dt + \sigma dZ(t)$$

単語の場合

t で状態 ① or ② どちらか分かる

サンプルパス

(単語が従う方程式や関数)が
分からぬ

最後に

10/31

問題点

- どうやってサンプルパスを推定するか
- サンプルパスと状態の比較をして何が言えるのか考える

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

参考にさせていただいた論文

主に [1] の再現を中心に行い、[2]～[4] は適宜離散化やアルゴリズム実装時に参考させていただきました。

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

[1] 長江剛志, 赤松隆: 連鎖的な意思決定構造を持つプロジェクトの動学的評価法: オプション・グラフ・モデルとその解法, 土木学会論文集, No. 772/IV-65, pp. 185–202, 2004

[2] 赤松隆・長江剛志: 社会基盤整備・運用事業の経済リスク管理問題に対するファイナンス工学的アプローチ, 土木計画学研究・論文集, No. 23, pp. 1-21, 2006

[3] Akamatsu, T. and Nagae, T.: A network of options: Evaluating complex interdependent decisions under uncertainty, *Journal of Economic Dynamics and Control*, Vol. 35, pp. 714-729, 2011.

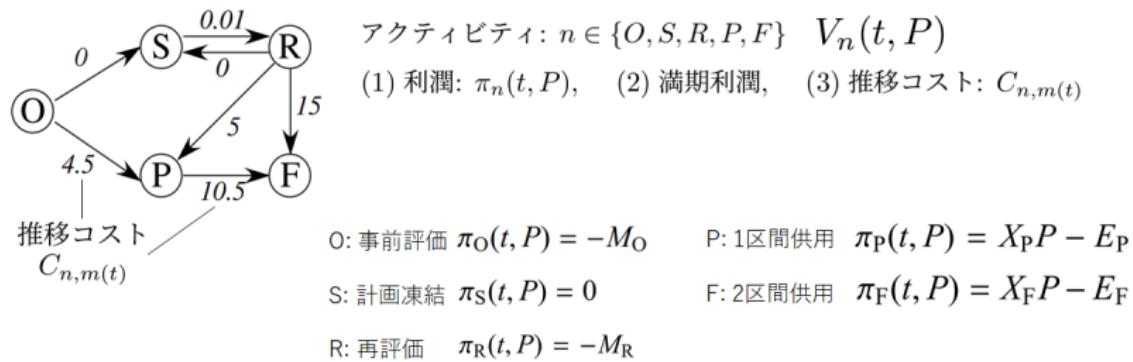
[4] Nagae, T. and Akamatsu, T.: Dynamic Revenue Management of Toll Road Projects under Transportation Demand Uncertainty, *Networks and Spatial Economics*, Volume 6, Numbers 3-4, pp. 345-357, 2006.

現在行っていること

12/31

[1] 8 章の例題の再現を (12p アルゴリズムを用いて) 行っております

研究背景
既存研究
提案手法
離散化
アルゴリズムの再現
偏微分方程式の差分化



$$T = 20, r = 5\%, \alpha = 1\%, \sigma = 40\%$$

$$M_O = 0.02, M_R = 0.01$$

$$X_P = 0.5, E_P = 0.6, X_F = 1, E_F = 1$$

離散化

時間 t と状態 P の離散化

- 以下の設定
- 「 P がどの i においても P_0 から $P_0 + J\Delta P$ まで変化する」と仮定

$$(t^i, P^j) = (i\Delta T, P_{\min} + j\Delta P)$$

$$(i = 0, 1, \dots, I), (j = 0, 1, \dots, J)$$

$$I = J = 400$$

$$\Delta T = \frac{T - 0}{400} = \frac{1}{20}$$

$$\Delta P = \frac{P^J - P_0}{400} = \frac{1}{50}$$

時間: $t \in [0, T]$ ($T = 20$)

状態空間: $[P_{\min}, P_{\max}] \in \mathcal{R}$

$$P_{\min} = P_0 = 0$$

$$P_{\max} = P^J = 8$$

懸念 1.

$\Delta T, \Delta P$ の設定について

$$P^j = P_0 + j\Delta P$$

P^j	$j = 0$	$j = 1$	$j = 2$	\dots	$j = J$
$i = 0$					
\vdots	P_0	$P_0 + \Delta P$	$P_0 + 2\Delta P$	\dots	$P_0 + J\Delta P$
$i = I$					

離散化

懸念 2.

P^j 縮小化の設定について

研究背景
既存研究
提案手法
縮小化
アルゴリズムの再現

偏微分方程式の差分化

$$P^j = P_0 + j\Delta P$$

$$= \begin{array}{|c|c|c|c|c|c|} \hline P^j & j = 0 & j = 1 & j = 2 & \cdots & j = J \\ \hline i = 0 & P_0 & P_0 + \Delta P & P_0 + 2\Delta P & \cdots & P_0 + J\Delta P \\ \hline \vdots & & & & & \\ \hline i = I & & & & & \\ \hline \end{array} \quad \text{---(※)}$$

$P(t)$ は幾何ブラウン運動に従う ([1], p15)

$$dP(t)/P(t) = \alpha dt + \sigma dZ(t) \quad \text{---(※※)}$$

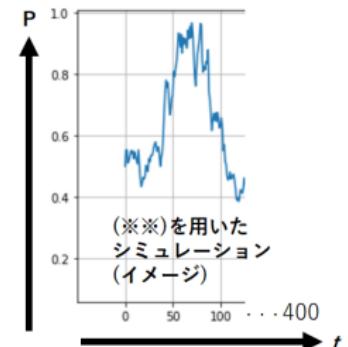
懸念 2.

(※) のように変化する場合,

P^j は幾何ブラウン運動 (※※) に従わないと形になると思われる

2.1. ドリフト・拡散項を使用する意味の記述であるのか?

$$\begin{aligned} \alpha(t, P, n) &= \alpha P \\ \sigma(t, P, n) &= \sigma P \end{aligned} \quad [2]$$



2.2. ※※をプログラムでシミュレートして得られる「[0, 8]」で変化する $P(t)$ を用いることを意味した記述か?

- 2.1. の想定で $\alpha(t, P)$, $\sigma(t, P)$, $\pi_n^i(t, P)$ を離散化

$$\alpha(t, P) = \alpha P$$

αP^j	$j = 0$	$j = 1$	$j = 2$	\cdots	$j = J$
$i = 0$	αP_0	$\alpha(P_0 + \Delta P)$	$\alpha(P_0 + 2\Delta P)$	\cdots	$\alpha(P_0 + J\Delta P)$
\vdots					
$i = I$					

$$\sigma(t, P) = \sigma P \simeq \sigma P^j \text{ (同様)}$$

$$\pi_F(t, P) = P - 1$$

$$\simeq \boldsymbol{\pi}_F^i = \pi(t^i, P^j) = P^j - \mathbf{1} = \begin{bmatrix} P_0 - 1 \\ P_0 + \Delta P - 1 \\ P_0 + 2\Delta P - 1 \\ \vdots \\ P_0 + J\Delta P - 1 \end{bmatrix}^{(J \text{ 次元})}$$

$$\boldsymbol{\pi}_P^i = 0.5P^j - 0.6 \cdot \mathbf{1}$$

$$\boldsymbol{\pi}_S^i = \mathbf{0}$$

$$\boldsymbol{\pi}_R^i = -0.01 \cdot \mathbf{1}$$

$$\boldsymbol{\pi}_O^i = -0.02 \cdot \mathbf{1}$$

アルゴリズムの再現

[1] 12p のアルゴリズムを再現し, "O", "S", "R", "P", "F" の価値を導出

研究背景
既存研究
提案手法
離散化
アルゴリズムの再現

偏微分方程式の差分化

[Alg-Option Graph]

(終端条件)

for all $n \in N$ do

$$V_n^i := \max \left[F_n, \max_{m \in O(n)} \left(V_m^i - 1C_{n,m} \right) \right];$$

end for

(時刻についての逆向き帰納法)

for $i := 1 - 1$ to 0 step -1 do

(初期化)

for all $n' \in N^E$ do

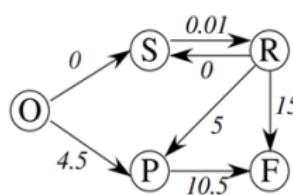
差分方程式 (28) を解いて $V_{n'}^i$ を求める;

end for

$A(n) := |O(n)|$ for all $n \in N$;

$B(u) := |O(u)|$ for all $c \in C$;

$\hat{N} := N^E$



(グラフ構造ごとに分解された問題の求解)

while $\hat{N} \neq \emptyset$ do

\hat{N} の先頭要素を n とし, n を \hat{N} から取り除く;

for all $m \in I(n)$ do

$A(m) := A(m) - 1$

if $A(m) = 0$ then

[Alg-Merit] を用いて問題 $[LCP^{i-m}]$ を解き, V_m^i を求める;

m を \hat{N} の末尾に挿入する;

end if

end for

for all $c \in I_C(n)$ do

$B(c) := B(c) - 1$

if $B(c) = 0$ then

[Alg-Merit] を用いて問題 $[NCP^{i-c}]$ を解き, V_c^i を求める;

N_c の全要素を \hat{N} の末尾に挿入する;

end if

end for

end while

end for

終端条件について

懸念 3.

終端条件 $\mathbf{V}_F^I, \mathbf{V}_P^I, \mathbf{V}_R^I, \mathbf{V}_S^I, \mathbf{V}_O^I$ は全て 0 であるか?
(終端ペイオフは, $F_n = 0$ と仮定 ([3] 7 章))

(終端条件)

for all $n \in N$ **do**

$$V_n^I := \max. \left[F_n, \max_{m \in O(n)} \{ V_m^I - \mathbf{1}C_{n,m} \} \right];$$

end for

---(※※※)

3.1. [2](p12) では $\mathbf{V}_n^i = \mathbf{0}$

3.2. [1],[3] では \mathbf{V}_n^i の明記がみられなかったが, (※※※) を用いて求める形か?
→計算済み $m \in O(n)$ の価値を用いるため,

相互に価値未決定のサイクル構造 $(\mathbf{V}_R^I, \mathbf{V}_S^I)$ の計算不可?

現状 $\mathbf{V}_n^I = \mathbf{0}$ として実装

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

[1] に従い、偏微分方程式を離散化

価値 V を求める式 (終端アクティビティの場合)

$$\frac{\partial V(t, P)}{\partial t} + \alpha(t, P) \frac{\partial V(t, P)}{\partial P} + \frac{1}{2} \{\sigma(t, P)\}^2 \frac{\partial^2 V(t, P)}{\partial P^2} - rV(t, P) + \pi(t, P) = 0$$

差分化 (Crank-Nicolson法) [4]

$$\begin{bmatrix}
 b_1^i & c_1^i & 0 & 0 & \cdots & 0 \\
 a_2^i & b_2^i & c_2^i & 0 & \cdots & 0 \\
 0 & a_3^i & b_3^i & c_3^i & \cdots & 0 \\
 \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 0 & \cdots & a_{J-3}^i & b_{J-3}^i & c_{J-3}^i & 0 \\
 0 & \cdots & 0 & a_{J-2}^i & b_{J-2}^i & c_{J-2}^i \\
 0 & \cdots & 0 & 0 & a_{J-1}^i & b_{J-1}^i
 \end{bmatrix}
 \begin{bmatrix}
 V_1^i \\
 V_2^i \\
 V_3^i \\
 \vdots \\
 V_{J-3}^i \\
 V_{J-2}^i \\
 V_{J-1}^i
 \end{bmatrix}
 +
 \begin{bmatrix}
 d_1^i & c_1^i & 0 & 0 & \cdots & 0 \\
 a_2^i & d_2^i & c_2^i & 0 & \cdots & 0 \\
 0 & a_3^i & d_3^i & c_3^i & \cdots & 0 \\
 \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
 0 & \cdots & a_{J-3}^i & d_{J-3}^i & c_{J-3}^i & 0 \\
 0 & \cdots & 0 & a_{J-2}^i & d_{J-2}^i & c_{J-2}^i \\
 0 & \cdots & 0 & 0 & a_{J-1}^i & d_{J-1}^i
 \end{bmatrix}
 \begin{bmatrix}
 V_1^{i+1} \\
 V_2^{i+1} \\
 V_3^{i+1} \\
 \vdots \\
 V_{J-3}^{i+1} \\
 V_{J-2}^{i+1} \\
 V_{J-1}^{i+1}
 \end{bmatrix}
 +
 \begin{bmatrix}
 \pi_1^i \\
 \pi_2^i \\
 \pi_3^i \\
 \vdots \\
 \pi_{J-3}^i \\
 \pi_{J-2}^i \\
 \pi_{J-1}^i
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 0 \\
 \vdots \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

\mathbf{L}_n^i ($J \times J$ 次元) \mathbf{V}_n^i (J 次元) \mathbf{M}_n^i ($J \times J$ 次元) \mathbf{V}_n^{i+1} (J 次元)

行列内の要素 [4]

$$\begin{aligned}
 a_j^i &= -\frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2} \\
 b_j^i &= -\frac{1}{\Delta T} - \frac{(\sigma_j^i)^2}{2(\Delta P)^2} - r \\
 c_j^i &= \frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2} \\
 d_j^i &= \frac{1}{\Delta T} - \frac{(\sigma_j^i)^2}{2(\Delta P)^2} \quad (\alpha_j^i = \alpha P^j) \\
 &\quad (\sigma_j^i = \sigma P^j)
 \end{aligned}$$

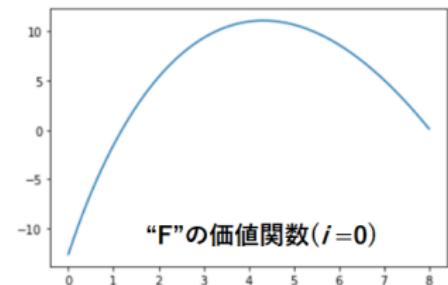
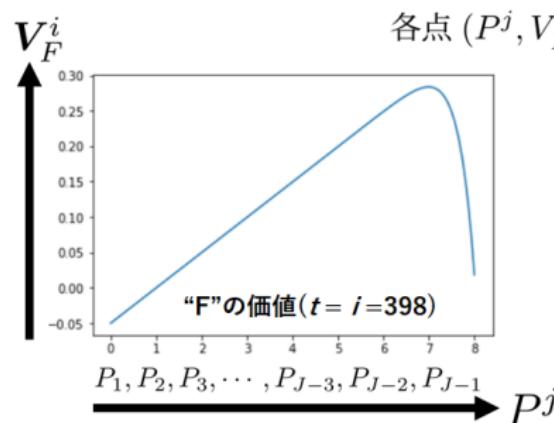
$$\mathbf{L}^i \begin{bmatrix} V_1^i \\ V_2^i \\ V_3^i \\ \vdots \\ V_{J-3}^i \\ V_{J-2}^i \\ V_{J-1}^i \end{bmatrix} + \mathbf{M}^i \begin{bmatrix} V_1^{i+1} \\ V_2^{i+1} \\ V_3^{i+1} \\ \vdots \\ V_{J-3}^{i+1} \\ V_{J-2}^{i+1} \\ V_{J-1}^{i+1} \end{bmatrix} + \boldsymbol{\pi}^i = \mathbf{0}$$

$i = I \rightarrow i = 0$:
 \mathbf{V}_n^i 逐次導出

V_F^i の結果確認

先に全時刻で V_F^i を出力

```
for i in range(I-1):
    # 終端アクティビティFの価値計算
    i_rev = I-1-i
    act_now["F"][i_rev-1,:] = ( -np.linalg.inv(L[i_rev-1])
    | @ (M[i_rev-1] @ act_now["F"][i_rev,:].T + act_pi["F"][i_rev-1,:].T ) .T
```



懸念 4. [1] 各図は exp 型の増加だが、結果は増加→減少型の曲線となったこと

Merit 関数による $\mathbf{X}_n^i(V_n^i)$ の更新

【解釈】最適 $\mathbf{X}_c^i (\mathbf{X}_n^i)$ を求めることで、(※) から \mathbf{V}_n^i が求められる

研究背景

既存研究

提案手法

離散化

アルゴリズムの

再現

偏微分方程式の差
分化

求めたいもの ([LCPⁱ-n] も同様)

[NCPⁱ-c] Find \mathbf{X}_c^i such that

$$\mathbf{X}_c^i \cdot \mathbf{G}_c^i(\mathbf{X}_c^i) = 0, \quad \text{and} \quad \mathbf{X}_c^i \geq \mathbf{0}, \quad \mathbf{G}_c^i(\mathbf{X}_c^i) \geq \mathbf{0}.$$

[Alg-Merit]

Step 0 初期可能解 $\mathbf{X}_c^{i(1)} \in \mathcal{R}_+$, $k := 1$.

Step 1 降下方向ベクトルの決定.

$$\mathbf{d}^{(k)} := \mathbf{H}(\mathbf{X}_c^{i(k)}). \quad (34)$$

Step 2 ステップ・サイズ α を、以下の一次元探索問題の解として求める.

$$\alpha^* = \arg \min_{\alpha \in [0, 1]} \Phi(\mathbf{X}_c^{i(k)} + \alpha \mathbf{d}^{(k)}). \quad (35)$$

Step 3 解の改訂. $\mathbf{X}_c^{i(k+1)} := \mathbf{X}_c^{i(k)} + \alpha^* \mathbf{d}^{(k)}$

Step 4 収束判定: 収束していれば停止、そうでなければ $k := k + 1$ として Step 1 へ.

解釈

$\mathbf{G}(\mathbf{X}_c^i)$ を求める処理

$$\mathbf{L}_n^i \mathbf{V}_n^{i(k)} = -\mathbf{X}_n^{i(k)} - \mathbf{M}_n^i \mathbf{V}_n^{i+1} - \boldsymbol{\pi}_n^i \quad \text{----- (※)}$$

$$\begin{aligned} \mathbf{G}_n^{i(k)} &:= \min \left[\min_{m \in O_c(n)} \left\{ \mathbf{V}_n^{i(k)} - \mathbf{V}_m^{i(k)} - \mathbf{1} \mathbf{C}_{n,m} \right\}, \right. \\ &\quad \left. \min_{m' \in O_c(n)} \left\{ \mathbf{V}_n^{i(k)} - \mathbf{V}_{m'}^{i(k)} - \mathbf{1} \mathbf{C}_{n,m'} \right\} \right], \quad \forall n \in N_c. \end{aligned}$$

Step1: $\mathbf{H}(\mathbf{X}_c^i)$ を求める処理

$$\mathbf{H}(\mathbf{X}_c^i) \equiv \left[\mathbf{X}_c^i - \mathbf{G}(\mathbf{X}_c^i) \right]_+ - \mathbf{X}_c^i$$

Step2: $\Phi(\mathbf{X}_c^i)$ を求める処理

$$\Phi(\mathbf{X}_c^i) \equiv -\mathbf{G}(\mathbf{X}_c^i) \cdot \mathbf{H}(\mathbf{X}_c^i) - \frac{1}{2} \mathbf{H}(\mathbf{X}_c^i) \cdot \mathbf{H}(\mathbf{X}_c^i)$$

Merit 関数の実装

$H(\mathbf{X}_n^i)$, $G(\mathbf{X}_n^i)$, $\Phi(\mathbf{X}_n^i)$ を呼び出し関数として実装

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

・ $H(\mathbf{X}_n^i)$ を求める処理

$$H(\mathbf{X}_n^i) = \left[\mathbf{X}_n^i - G(\mathbf{X}_n^i) \right]_+ - \mathbf{X}_n^{i(k)}$$

→【解釈】要素 $X^{i,j}$ は正または 0

・ $G(\mathbf{X}_n^i)$ を求める処理

$$L_n^i V_n^{i(k)} = -\mathbf{X}_n^{i(k)} - M_n^i V_n^{i+1} - \pi_n^i \quad \text{サイクル構造の場合}$$

$$G_n^{i(k)} = \min \left[\min_{m \in O_c(n)} \{ V_n^{i(k)} - V_m^{i(k)} - \mathbf{1} C_{n,m} \}, \min_{m' \in \hat{O}_c(n)} \{ V_n^{i(k)} - V_{m'}^{i(k)} - \mathbf{1} C_{n,m'} \} \right]$$

懸念 5. ベクトルの最小比較について

J 次元の各要素どうしで最小比較されると仮定

例) アクティビティ "O" の場合, $\min \left[V_O^{i(k)} - V_S^{i(k)} - \mathbf{1} C_{O,S}, V_O^{i(k)} - V_P^{i(k)} - \mathbf{1} C_{O,P} \right]$

$$V_O^{i(k),0} - V_S^{i(k),0} - C_{O,S} \quad j = 0$$

$$V_O^{i(k),1} - V_P^{i(k),1} - C_{O,P} \quad j = 1$$

$$G_O^{i(k)} = V_O^{i(k),2} - V_P^{i(k),2} - C_{O,P} \quad j = 2$$

⋮

$$V_O^{i(k),J} - V_S^{i(k),J} - C_{O,S} \quad j = J$$

"S" と "P" の要素が入り混じると想定

Merit 関数の実装

[Alg-Merit] を以下のように解釈して実装

・ $\Phi(X_n^i)$ を求める処理

$$\Phi(\mathbf{X}_n^i) = -\mathbf{G}(\mathbf{X}_n^i) \cdot \mathbf{G}(\mathbf{X}_n^i)^\top - \frac{1}{2}\mathbf{H}(\mathbf{X}_n^i) \cdot \mathbf{H}(\mathbf{X}_n^i)^\top$$

Step0: 初期状態 $\mathbf{X}_n^{i(1)} = \mathbf{0}$ (J 次元ベクトル)

Step1: $\mathbf{d}^{(k)} = \mathbf{H}(\mathbf{X}_n^i)$

Step2: $\alpha \in [0, 1] \rightarrow \alpha = \frac{l}{1000}, (l = 0, 1, \dots, 1000)$

for l (0 to 1000): $G(\mathbf{X}_n^{i(k)} + \frac{l}{1000}\mathbf{d}^{(k)}) \rightarrow H(\mathbf{X}_n^{i(k)} + \frac{l}{1000}\mathbf{d}^{(k)})$
 $\rightarrow \Phi(\mathbf{X}_n^{i(k)} + \frac{l}{1000}\mathbf{d}^{(k)})$ 導出 \rightarrow リストに格納

リスト中で $\Phi(X_n^{i(k)} + \frac{l}{1000}d^{(k)})$ が最小のインデックス l^* を求める

Step3: $\mathbf{X}_n^{i(k+1)} = \mathbf{X}_n^{i(k)} + \frac{l^*}{1000} \mathbf{d}^{(k)}$

Step4: $X_n^{i(k+1)} \equiv X_n^{i(k)}$ なら停止

それ以外は $k = k + 1$

懸念 6. 収束判定について

$$X_n^{i(k+1)} \succcurlyeq X_n^{i(k)}$$

J 個の要素が全て一致する必要があるのか？

Merit 関数の実装 (コーディング)

V_F^i から V_P^i を求める過程を実装 (Python)

研究背景

既存研究

提案手法

離散化

アルゴリズムの再現

偏微分方程式の差分化

Gx処理定義 (Python)

```

1 # G(X,n)*を求める処理(併サイクル構造)
2 def getGxRowForX(X, n):
3     act_v[n][i_rev-1,:] = -(np.linalg.inv([i]) @ (X.T + M[i]) @ act_v[n][i_rev-1,:].T
4     + act_pi[n][i_rev-1,:].T).T
5     On = list(act_trans[n])
6     On_k2 = []
7     for ix in On:
8         Go_ks.append(act_v[n][i_rev-1,:] - act_v[ix][i_rev-1,:]
9                     - np.matrix(np.full((1,i), act_cost[n][ix]), dtype='float'))
10    return np.minimum.reduce(Go_ks)
11
12 # G(X,n)*を求める処理(サクル構造)
13 def getGxRowForXc(Xc, nc):
14     act_vc[n][i_rev-1,:] = -(np.linalg.inv([i]) @ (Xc.T + M[i]) @ act_vc[n][i_rev-1,:].T
15     + act_pi[n][i_rev-1,:].T).T
16     act_vc[n][1][i_rev-1,:] = -(np.linalg.inv([i]) @ (Xc.T + M[i]) @ act_vc[n][1][i_rev-1,:].T
17     + act_pi[n][1][i_rev-1,:].T).T
18
19     On = list(act_trans[nc[0]])
20     On2 = list(act_trans[nc[1]])
21
22     Go_ks = []
23     for ix in On:
24         Go_ks.append(act_vc[n][0][i_rev-1,:] - act_vc[ix][i_rev-1,:]
25                     - np.matrix(np.full((1,i), act_cost[nc[0]][ix]), dtype='float'))
26
27     Go_k2s = []
28     for iy in On2:
29         Go_k2s.append(act_vc[n][1][i_rev-1,:] - act_vc[iy][i_rev-1,:]
30                     - np.matrix(np.full((1,i), act_cost[nc[1]][iy]), dtype='float'))
31
32     Go_k2s.append(np.minimum.reduce(Go_ks))
33
34     return np.minimum.reduce(Go_k2s)

```

Merit関数 (Python)

```

33 for i in range(1-1):
34     i_rev = 1-1
35     act_v["*"][[i_rev-1,:]] = (-np.linalg.inv([i][i_rev-1]))
36     # (M[i_rev-1] @ act_v["*"][[i_rev-1,:].T + act_pi["*"][[i_rev-1,:].T]).T
37
38     ##### vjを求める処理 #####
39     x0_ik = np.matrix(np.full((1,i)), 0), dtype='float')
40     d_k = np.matrix(np.full((1,i), 0), dtype='float')
41
42     k = 1
43     isExist = True
44     while isExist: # Xn_ikを更新される限りを循環
45         ###### Step1 #####
46         d_k = np.maximum(Xn_ik - getGxRowForXc(Xn_ik, "p"),
47                           np.matrix(np.full((1,i), 0)), dtype='float') - x0_ik
48         ###### Step2 #####
49         phi = []
50         L = 1000 # 1000
51         for l in range(L+1):
52             Xn_re = Xn_ik + (1 / L) * d_k
53             G_X = getGxRowForXc(Xn_re, "p")
54             Xn_re = np.matrix(Xn_re - G_X, dtype='float') - Xn_re
55             phi += G_X @ Xn_re.T * (1 / L) + x0_ik @ Xn_re.T
56             phi.append(phi[-1]) # リストに格納
57             als = np.argmax(phi) # 小さな方を抜く
58             als = np.argmax(phi) # 小さな方を抜く
59         ###### Step3 #####
60         Xn_ik_re = Xn_ik + (als / L) * d_k
61         ###### Step4 #####
62         if (Xn_ik_re == Xn_ik).all():
63             isExist = False # 収束したので終止条件
64         else:
65             k = k + 1
66             Xn_ik = Xn_ik_re

```

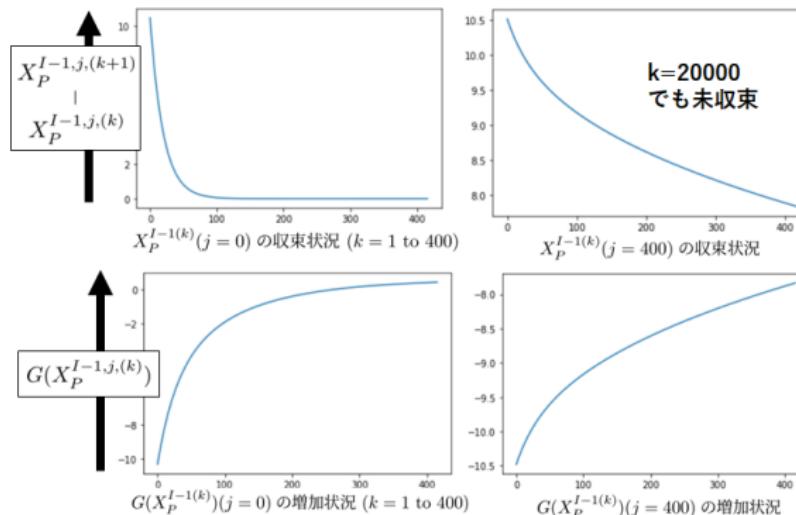
$$I = J = 400$$

あらかじめ全時刻 i で \mathbf{V}_F^i から \mathbf{V}_P^i のみを計算

\mathbf{X}_P^{I-1} の時点で収束しなかった ($\rightarrow k = 400$ (約 1 時間半) 経過)

起こっている現象

- どの i でも, $\alpha^* = 1 (l^* = 1000)$ が選ばれる
- \mathbf{X}_P^i の要素 ($j = 390$ まで) は収束ぎみ, 391 以降は時間がかかる
- $G(\mathbf{X}_P^i)$ の要素 ($j = 390$ まで) は ≈ 0 , 391 以降はゆっくり増加 (負 $\rightarrow 0$)



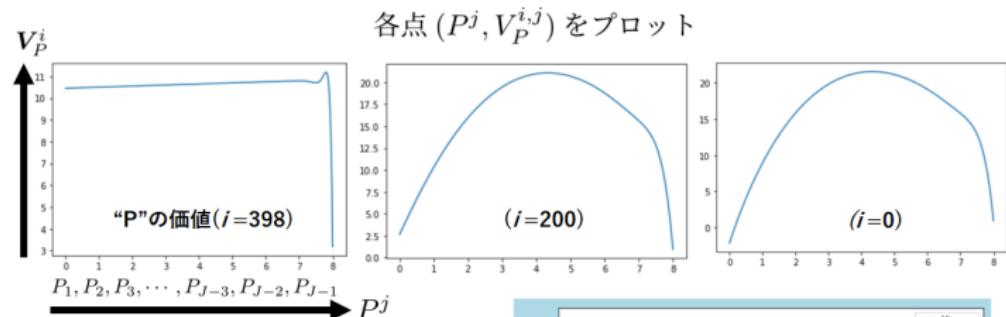
問題 ①

全時刻 i で \mathbf{X}_n^i を収束させ, $\mathbf{V}_F^i, \mathbf{V}_P^i, \mathbf{V}_R^i, \mathbf{V}_S^i, \mathbf{V}_O^i$ を求めたい

各種考察

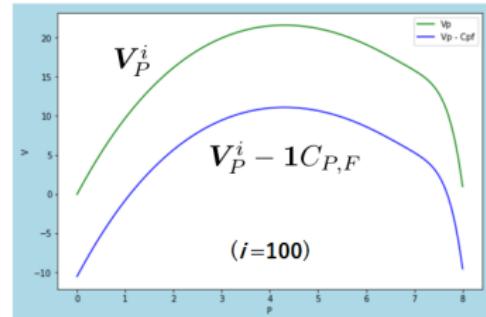
- ※ 前ページ = 収束判定「J 個の要素全て一致」の場合
- 初期値: $X_n^{i(1)} = -10$ や $X_n^{i(1)} = 100$ でも収束状況は変わらなかった

収束判定「少なくとも要素 1 つが一致」の場合でも適切な結果でなかった (下図)



$i = 0: k = 653,$
 $i = 1 \sim 10: k = 100 \sim 50,$
 i=10 以降: 常に $k = 63$ で収束

V_P^i を全て計算できたが,
 $V_P^i - 1C_{P,F}$ と接しなかった



研究背景

既存研究

提案手法

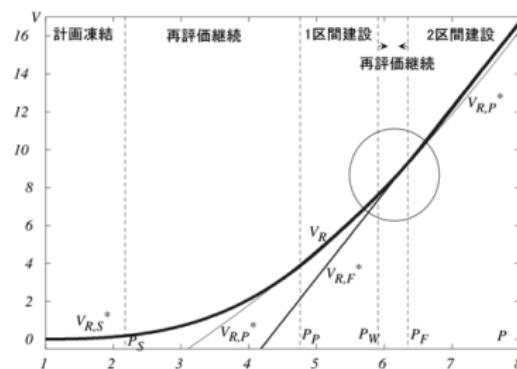
離散化

アルゴリズムの
再現

偏微分方程式の差
分化

懸念 4(再).

[1] 各図は exp 型の増加だが、
結果の V_F^i , V_P^i が増加→減少の曲線になっている点



"R" の価値関数 $t=15$ ($\Leftarrow i=300$)

問題②

[1] の各図と同様な形
($V_n^i - 1C_{n,m}$ と V_n^i が接するような
形と思われる) の再現

問題の解決

27/31

問題の解決

- 問題 1 と 2 を解決するために 6 つの懸念や各所の解釈違いが関わっていると考えられる

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の
分化

[1](28) 式の導出

- Crank-Nicholson 法による差分化→行列化の手順を確認

研究背景

既存研究

提案手法

離散化

アルゴリズムの
再現

偏微分方程式の差
分化

$$\frac{\partial V(t, P)}{\partial t} + \alpha(t, P) \frac{\partial V(t, P)}{\partial P} + \frac{1}{2} \{ \sigma(t, P) \}^2 \frac{\partial^2 V(t, P)}{\partial P^2} - rV(t, P) + \pi(t, P) = 0$$



差分化 (Crank-Nicolson法)

$$[4] \frac{V_j^{i+1} - V_j^i}{\Delta T} + \frac{1}{2} \alpha_j^i \left(\frac{V_{j+1}^i - V_{j-1}^i}{2\Delta P} + \frac{V_{j+1}^{i+1} - V_{j-1}^{i+1}}{2\Delta P} \right) + \frac{1}{2} \frac{(\sigma_j^i)^2}{2} \left(\frac{V_{j+1}^i - 2V_j^i + V_{j-1}^i}{(\Delta P)^2} + \frac{V_{j+1}^{i+1} - 2V_j^{i+1} + V_{j-1}^{i+1}}{(\Delta P)^2} \right) - rV_j^i + \pi_j^i = 0$$



$$\left(-\frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2} \right) V_{j-1}^i + \left(-\frac{1}{\Delta T} - \frac{(\sigma_j^i)^2}{2(\Delta P)^2} - r \right) V_j^i + \left(\frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2} \right) V_{j+1}^i + \left(-\frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2} \right) V_{j-1}^{i+1} + \left(\frac{1}{\Delta T} - \frac{(\sigma_j^i)^2}{2(\Delta P)^2} \right) V_j^{i+1} + \left(\frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2} \right) V_{j+1}^{i+1} + \pi_j^i = 0$$

項の置き換え

$\underline{a_j^i} = -\frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2}$	$\underline{c_j^i} = \frac{\alpha_j^i}{4\Delta P} + \frac{(\sigma_j^i)^2}{4(\Delta P)^2}$	$(\alpha_j^i = \alpha P^j)$
$\underline{b_j^i} = -\frac{1}{\Delta T} - \frac{(\sigma_j^i)^2}{2(\Delta P)^2} - r$	$\underline{d_j^i} = \frac{1}{\Delta T} - \frac{(\sigma_j^i)^2}{2(\Delta P)^2}$	$(\sigma_j^i = \sigma P^j)$

[1](28) 式の導出



$$a_j^i V_{j-1}^i + b_j^i V_j^i + c_j^i V_{j+1}^i + a_j^i V_{j-1}^{i+1} + d_j^i V_j^{i+1} + c_j^i V_{j+1}^{i+1} + \pi_j^i = 0 \quad (1)$$

$a_j^i V_{j-1}^i + b_j^i V_j^i + c_j^i V_{j+1}^i$ について $j = 1$ から $j = J - 1$ まで展開

$$j = 1 \quad a_1^i V_0^i + b_1^i V_1^i + c_1^i V_2^i$$

$$j = 2 \quad a_2^i V_1^i + b_2^i V_2^i + c_2^i V_3^i$$

$$j = 3 \quad a_3^i V_2^i + b_3^i V_3^i + c_3^i V_4^i$$

⋮

⋮

$$j = J - 3 \quad a_{J-3}^i V_{J-4}^i + b_{J-3}^i V_{J-3}^i + c_{J-3}^i V_{J-2}^i$$

$$j = J - 2 \quad a_{J-2}^i V_{J-3}^i + b_{J-2}^i V_{J-2}^i + c_{J-2}^i V_{J-1}^i$$

$$j = J - 1 \quad a_{J-1}^i V_{J-2}^i + b_{J-1}^i V_{J-1}^i + c_{J-1}^i V_J^i$$

[1](28) 式の導出



研究背景

既存研究

提案手法

離散化

アルゴリズムの

再現

偏微分方程式の差
分化

同様の V_j^i をまとめることで行列化

$$\text{for } j \text{ in } (1 \leq j \leq J-1) \quad a_j^i V_{j-1}^i + b_j^i V_j^i + c_j^i V_{j+1}^i =$$

※ (1) 式 第 4 項以降も同様に展開

$$\begin{bmatrix} b_1^i & c_1^i & 0 & 0 & \cdots & 0 \\ a_2^i & b_2^i & c_2^i & 0 & \cdots & 0 \\ 0 & a_3^i & b_3^i & c_3^i & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{J-3}^i & b_{J-3}^i & c_{J-3}^i & 0 \\ 0 & \cdots & 0 & a_{J-2}^i & b_{J-2}^i & c_{J-2}^i \\ 0 & \cdots & 0 & 0 & a_{J-1}^i & b_{J-1}^i \end{bmatrix} \begin{bmatrix} V_1^i \\ V_2^i \\ V_3^i \\ \vdots \\ V_{J-3}^i \\ V_{J-2}^i \\ V_{J-1}^i \end{bmatrix} + \begin{bmatrix} a_1^i V_0^i \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ c_{J-1}^i V_J^i \end{bmatrix}$$

まとめると以下の式が得られる

$$\begin{bmatrix} b_1^i & c_1^i & 0 & 0 & \cdots & 0 \\ a_2^i & b_2^i & c_2^i & 0 & \cdots & 0 \\ 0 & a_3^i & b_3^i & c_3^i & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{J-3}^i & b_{J-3}^i & c_{J-3}^i & 0 \\ 0 & \cdots & 0 & a_{J-2}^i & b_{J-2}^i & c_{J-2}^i \\ 0 & \cdots & 0 & 0 & a_{J-1}^i & b_{J-1}^i \end{bmatrix} \begin{bmatrix} V_1^i \\ V_2^i \\ V_3^i \\ \vdots \\ V_{J-3}^i \\ V_{J-2}^i \\ V_{J-1}^i \end{bmatrix} + \begin{bmatrix} a_1^i V_0^i \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ c_{J-1}^i V_J^i \end{bmatrix} = \begin{bmatrix} V_1^{i+1} \\ V_2^{i+1} \\ V_3^{i+1} \\ \vdots \\ V_{J-3}^{i+1} \\ V_{J-2}^{i+1} \\ V_{J-1}^{i+1} \end{bmatrix} = \begin{bmatrix} \pi_1^i \\ \pi_2^i \\ \pi_3^i \\ \vdots \\ \pi_{J-3}^i \\ \pi_{J-2}^i \\ \pi_{J-1}^i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$a_j^i V_{j-1}^i + b_j^i V_j^i + c_j^i V_{j+1}^i$$

$$a_j^i V_{j-1}^{i+1} + d_j^i V_j^{i+1} + c_j^i V_{j+1}^{i+1}$$

懸念事項

- 別の項が出現したため、 \mathbf{V}^i を簡単に求められない
- $\mathbf{0}$ とみなし、論文と同じ形で計算したこと

研究背景
既存研究
提案手法
離散化
アルゴリズムの再現
偏微分方程式の差分化

$$[1](28)式 \quad \mathbf{L}^i \begin{bmatrix} V_1^i \\ V_2^i \\ V_3^i \\ \vdots \\ V_{J-3}^i \\ V_{J-2}^i \\ V_{J-1}^i \end{bmatrix} + \mathbf{M}^i \begin{bmatrix} V_1^{i+1} \\ V_2^{i+1} \\ V_3^{i+1} \\ \vdots \\ V_{J-3}^{i+1} \\ V_{J-2}^{i+1} \\ V_{J-1}^{i+1} \end{bmatrix} + \boldsymbol{\pi}^i = \mathbf{0}$$

懸念 7

- $\mathbf{0}$ とみなせる要因があるのか？
- 何かしらの近似が行われるのか？

$\mathbf{0}$ として計算

$$\text{自分で導出した式} \quad \mathbf{L}^i \begin{bmatrix} V_1^i \\ V_2^i \\ V_3^i \\ \vdots \\ V_{J-3}^i \\ V_{J-2}^i \\ V_{J-1}^i \end{bmatrix} + \begin{bmatrix} a_1^i V_0^i \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ c_{J-1}^i V_J^i \end{bmatrix} + \mathbf{M}^i \begin{bmatrix} V_1^{i+1} \\ V_2^{i+1} \\ V_3^{i+1} \\ \vdots \\ V_{J-3}^{i+1} \\ V_{J-2}^{i+1} \\ V_{J-1}^{i+1} \end{bmatrix} + \begin{bmatrix} a_1^{i+1} V_0^{i+1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ c_{J-1}^{i+1} V_J^{i+1} \end{bmatrix} + \boldsymbol{\pi}^i = \mathbf{0}$$