数理工学実験レポート

第 4 章（モンテカルロシミュレーション）

学籍番号 1029366161　　中塚一瑳

2025 年 12 月 2 日

概要

目次

1　課題 1：疫病の確率的な伝染モデル　2
1.1　原理と方法　2
1.2　実装上の工夫　2
1.3　小問 1　3
1.4　小問 2　4
2　課題 2：LU 分解　6
3　アルゴリズム比較　6
4　結論　6
付録 A　使用コード一覧　6

はじめに

今回は

1　課題 1：疫病の確率的な伝染モデル

1.1　原理と方法

本課題では，配布資料に記載の一次元確率的伝染モデルに従い，格子点 $i = 1, \ldots, 64$ に状態変数 $s_i(t) \in \{0, 1\}$ を配置した。左右近傍の感染者数を

$$n_i(t) = s_{i-1}(t) + s_{i+1}(t)$$

とし，次時刻の状態は資料式 (5.2) に従う確率 $P(1 \mid s_i(t), n_i(t))$ に基づいて決定する。周期境界条件は

$$s_0(t) = s_{64}(t), \qquad s_{65}(t) = s_1(t)$$

により与えた。

モンテカルロシミュレーションでは，時刻 t の状態列 $\{s_i(t)\}$ を初期状態 $t = 0$ から順に確率的に生成する。各格子点で一様乱数 $r \in [0, 1]$ を生成し，

$$s_i(t+1) = \begin{cases} 1 & (r \le P(1 \mid s_i(t), n_i(t))) \\ 0 & (r > P(1 \mid s_i(t), n_i(t))) \end{cases}$$

により次状態を決定した。

時刻 t の病人数は

$$m(t) = \sum_{i=1}^{64} s_i(t)$$

で定義し，独立な M 個のサンプルに対してサンプル平均

$$\overline{m(t)} = \frac{1}{M} \sum_{k=1}^{M} m^{(k)}(t)$$

を計算することで，期待値の近似を得た。

本課題では，初期条件として中央のみ感染（$s_{32}(0) = 1$）とし，シミュレーションの最終時刻は $T = 256$，サンプル数は M とした。

1.2　実装上の工夫

Python と NumPy を用いて実装を行った。Python で各格子点を for ループで逐次更新すると計算が極端に遅くなるため，本研究では全サイトをベクトル化により一括更新する方法を採用した。

まず，M 個の試行を同時に扱うために，状態を

$$s(t) \in \{0, 1\}^{M \times 64}$$

という 2 次元配列として保持した。周期境界条件に基づく左右近傍は NumPy の `roll` により

$$\texttt{right} = \texttt{roll}(s, -1), \qquad \texttt{left} = \texttt{roll}(s, 1)$$

として同時に生成し，$n(t) = \texttt{right} + \texttt{left}$ を一括して計算した。

資料式 (5.2) の遷移確率 $P(1 \mid s, n)$ は 6 通りの組 $(s, n) = (0,0), (0,1), (0,2), (1,0), (1,1), (1,2)$ に対応する値をあらかじめ一次元配列

$$\texttt{vals} = (\alpha_{00}, \alpha_{01}, \alpha_{02}, \alpha_{10}, \alpha_{11}, \alpha_{12})$$

として保持した。各格子点の組 (s_i, n_i) は

$$\texttt{index} = 3s + n$$

により 0–5 の整数に写像され,

$$\alpha = \texttt{vals[index]}$$

として全要素の遷移確率を分岐なしで取得した。

次状態の決定も同様にベクトル化し，乱数配列 $r \in [0,1]^{M \times 64}$ を生成して

$$s(t+1) = (r < \alpha)$$

という比較演算により全要素を同時更新した。病人数は `sum(axis=1)` を用いて

$$m^{(k)}(t) = \sum_i s_i^{(k)}(t)$$

を一括計算した。

以上のベクトル化により，明示的な for 文を用いず高速に M 試行 $\times$ 256 ステップのシミュレーションを実行できた。

1.3 小問 1

1.3.1 結果

パラメータ $p = 0.7$，サンプル数 $M = 10$ に対してシミュレーションを実行し, 各サンプルに対する 各時刻での m(t) を同時にプロットした（図 1）.

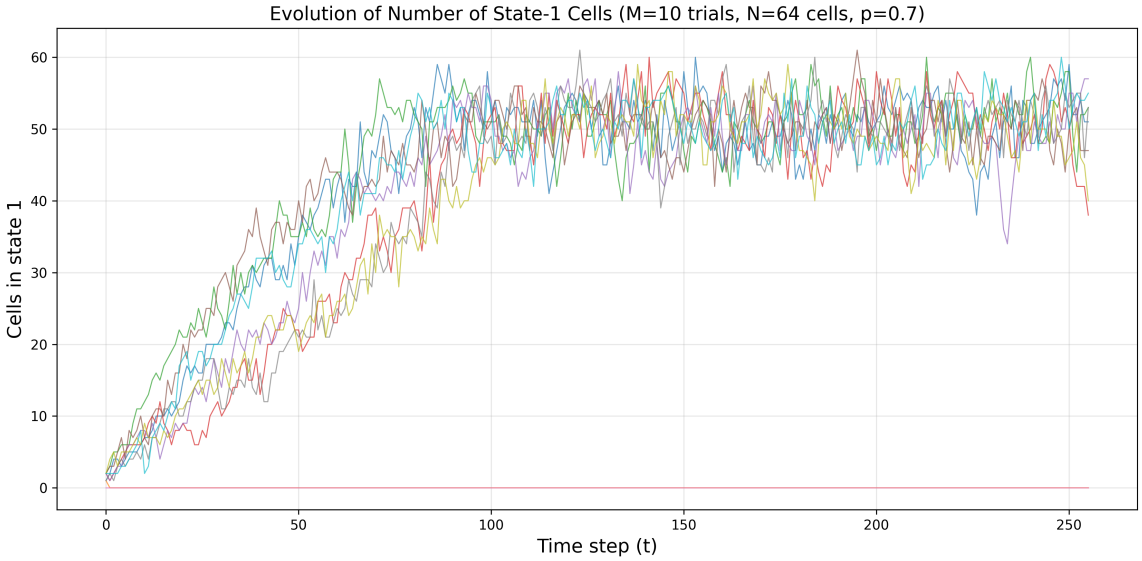

図 1: 各サンプルに対する 各時刻での $\mathrm{m(t)}$（$p = 0.7,\ M = 10$）

1.3.2 考察

図 1 に示すように, 各サンプルに対する 各時刻での $\mathrm{m(t)}$ は, 時間とともに増減を繰り返しながら増加し, 最終的には 50 前後を上下するようになっている. これは, 感染が広がるにつれて感染者数が増加する一方で, 回復によって感染者数が減少するつり合いの結果であると考えられる. また, ごく一部のサンプルでは, 開始直後に感染者が 0 となり, そのまま感染が広がらない場合も見られた. これは, 感染者が 0 となった時点で, その後の感染拡大が起こり得ないためであると考えられる.

1.4 小問 2

1.4.1 結果

パラメータ $p = 0.64, 0.66, 0.68, 0.7$, サンプル数 $M = 10000$ に対してシミュレーションを実行し, 各時刻での $\mathrm{m(t)}$ のサンプル平均をプロットした（図 2）.

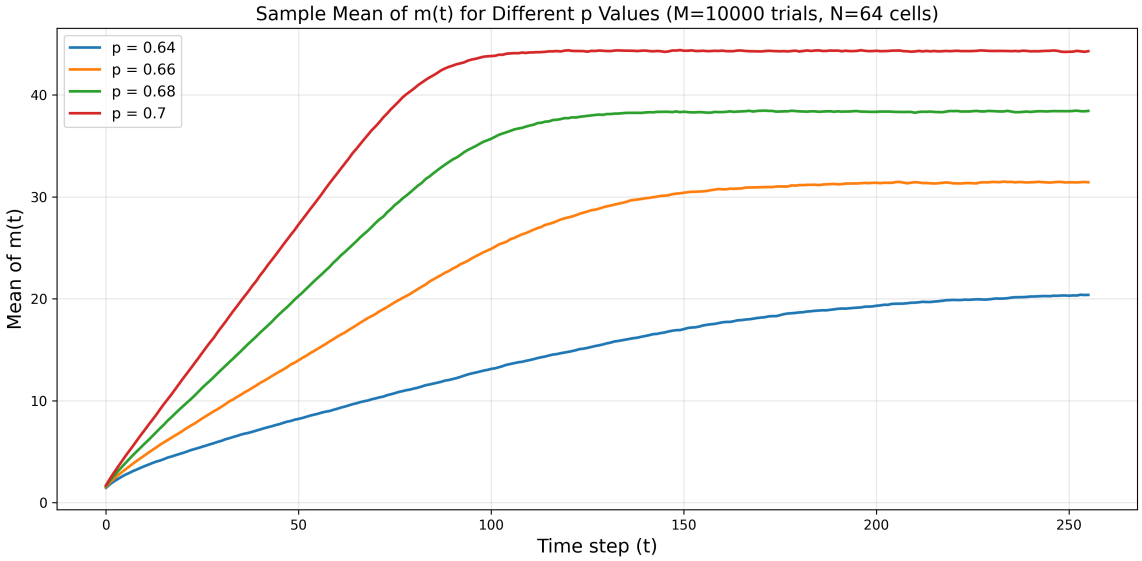

図 2: 各時刻での m(t) のサンプル平均（$p = 0.64, 0.66, 0.68, 0.7,\ M = 10000$）

1.4.2 考察

図 2 に示すように, 各時刻での m(t) のサンプル平均は, 時間とともに増加し, 最終的には一定の値に収束する傾向が見られた. また, パラメータ p が大きくなるにつれて, 最終的な感染者数の平均値も増加する傾向が見られた. これは, 感染確率が高いほど感染が広がりやすくなり, より多くの人々が感染するためであると考えられる. また、問題では最終時刻 256 としているが, T=100000 までシミュレーションを行った結果が以下の図である。

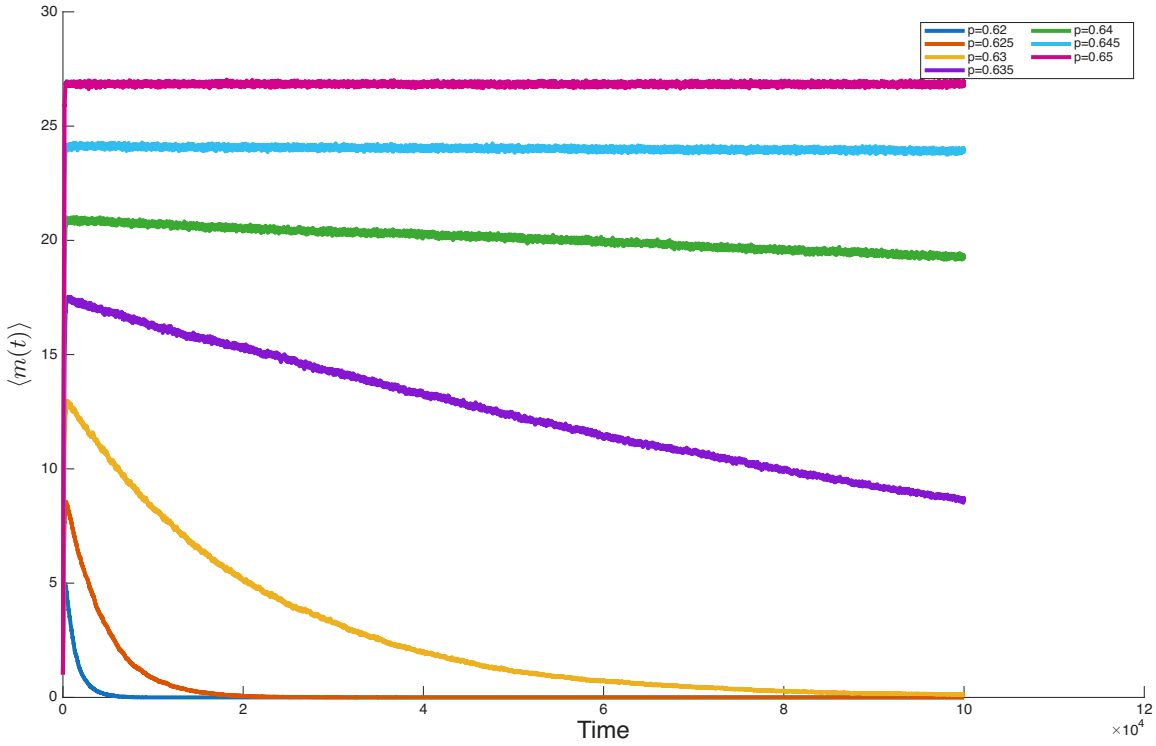

図 3: 各時刻での m(t) のサンプル平均（$p = 0.64, 0.66, 0.68, 0.7,\ M = 10000,\ T = 100000$）

図 3 が示すのは、$p = 0.65$ では一旦は増加した感染者数が一定の値を保っているのに対し、$p < 0.65$ では一旦は増加した感染者数が減少する方向に向かう。これは、$p = 0.65$ が臨界点であり、これを境に感染が広がるか収束するかが決定されるためであると考えられる。図 2 において $p = 0.64$ だけ収束が見えなかったのは、収束が遅いというよりも収束先が 0 であるためであるとも考えられる。

2　課題 2：LU 分解

目的：目的を記載

3　アルゴリズム比較

表や図を用いて複数手法の比較を行う。計算量・精度・安定性・実行時間の観点でまとめる。

4　結論

本章での結論と今後の課題を箇条書きでまとめる。

付録 A　使用コード一覧

主要スクリプトと入手先を列挙する。

参考文献

参考文献

[1] 数理工学実験（2025 年度配布資料）.