

博士論文

金融市場におけるマルチエージェント
シミュレーションとデータマイニングの融合

Fusion of Multi-agent Simulation and Data Mining
for Financial Markets

平野 正徳

目 次

第I部 初めに	1
第1章 研究概要	2
1.1 研究背景と目的	2
1.2 論文の構成と概要	4
第2章 関連研究	6
2.1 社会シミュレーション・マルチエージェントシミュレーションに関する研究	6
2.1.1 社会シミュレーションの発展と実例	6
2.1.2 人工市場シミュレーションに関する研究	7
2.1.3 シミュレーションの評価とパラメーター管理に関する研究	8
2.1.4 サロゲートモデルに関する研究	8
2.2 データマイニング・深層学習に関する研究	8
2.2.1 金融データマイニングに関する研究	9
2.2.2 敵対的生成ネットワーク (GAN)	9
2.2.3 強化学習の関連研究	10
2.3 人工市場シミュレーションと金融データマイニングの融合に関する研究	11
第II部 GAN の構築とシミュレーションの評価	12
第3章 PGSGAN: 金融市場における現実的な注文データ生成の方策勾配 GAN	13
3.1 本章における問題意識と背景	13
3.2 提案手法 1: Policy Gradient Stock GAN (PGSGAN)	16
3.2.1 PGSGAN の構造	16
3.2.2 PGSGAN の学習理論	18
3.2.3 PGSGAN の実際の実装	21
3.3 提案手法 2: Policy Gradient Stock GAN with Hinge Loss (PGSGAN-HL)	24
3.4 比較手法	24
3.4.1 Stock GAN (S-GAN)	24

3.4.2 DCGAN	25
3.5 実験	25
3.6 結果	33
3.7 考察	33
3.8 本章のまとめ	47
第4章 GANによるシミュレーション評価	48
4.1 シミュレーションの評価の難しさと研究背景	48
4.2 手法: GAN Evaluator	49
4.3 実験用モデルの実装	49
4.3.1 実験で使用したGANモデル(PGSGAN)	51
4.3.2 実験で使用したシミュレーションモデル	51
4.4 実験と評価	53
4.4.1 実験: パラメータチューニング	53
4.4.2 評価1: チューニング済みパラメータのアブレーションテスト	55
4.4.3 評価2: Stylized Factsの詳細分析	56
4.5 結果	56
4.5.1 実験: パラメータチューニング	56
4.5.2 評価1: チューニング済みパラメータのアブレーションテスト	57
4.5.3 評価2: Stylized Factsの詳細分析	59
4.6 考察	92
4.7 本章の結論	93
第III部 シミュレーションを用いたデータマイニング手法の評価	95
第5章 人工市場データマイニングプラットフォーム	96
5.1 金融データマイニングにおける問題点と本章の研究背景	96
5.2 人工市場データマイニングプラットフォーム	97
5.3 人工市場データマイニングプラットフォームによる実践	98
5.3.1 人工市場シミュレーションモデル	99
5.3.2 データマイニングモデル	101
5.4 実験	102
5.5 結果	103
5.6 考察	107
5.7 本章のまとめ	109
第6章 ケーススタディー：人工市場による注文同時性の効果分析	110
6.1 研究背景：金融市場の高速化と注文の同時性	110

6.2 モデル	111
6.2.1 人工市場シミュレーションモデル	112
6.2.2 データマイニングモデル: PGSGAN	112
6.2.3 仮説と検証手法	114
6.3 実験	115
6.4 結果	116
6.5 考察	117
6.6 本章のまとめ	120
第IV部 強化学習を用いたシミュレーションのチューニング	121
第7章 深層強化学習を用いたシミュレーションのパラメータチューニング	122
7.1 研究背景: マルチエージェントシミュレーションにおけるパラメータチューニングの重要性	122
7.2 提案手法	123
7.2.1 カスタム DDPG	125
7.2.2 カスタム SAC	127
7.2.3 Action Converter (AC)	128
7.2.4 Redundant Full Neural Network Actor (FNNNA)	129
7.2.5 Seed Fixer (SF)	130
7.3 実験	132
7.3.1 タスク設定	132
7.3.2 モデル	133
7.3.3 評価	134
7.4 結果	134
7.5 考察	141
7.6 本章のまとめ	143
第8章 GANによる評価と強化学習の融合	144
8.1 本章のモチベーションと目的	144
8.2 手法	145
8.3 実験	146
8.3.1 タスク設定	146
8.3.2 モデルの実装	147
8.3.3 比較評価	149
8.4 結果と考察	150
8.5 本章のまとめ	158

第V部　まとめ	159
第9章　全体を通した考察と今後の課題	160
第10章　まとめ	162
参考文献	177

図一覧

1.1	シミュレーションとデータマイニングのメリット・デメリット	3
1.2	シミュレーションとデータマイニングのメリット・デメリットの相補的関係	4
1.3	研究概要	5
3.1	PGGAN の概要	17
3.2	PGGAN の Generator の学習を強化学習の観点から示した概要	19
3.3	詳細な Generator の機構	22
3.4	詳細な Critic の機構	23
3.5	実データと生成された注文の分布の比較 (5901 JP)	34
3.6	実データと生成された注文の分布の比較 (5333 JP)	35
3.7	実データと生成された注文の分布の比較 (8355 JP)	36
3.8	実データと生成された注文の分布の比較 (5631 JP)	37
3.9	実データと生成された注文の分布の比較 (9532 JP)	38
3.10	実データと生成された注文の分布の比較 (7012 JP)	39
3.11	実データと生成された注文の分布の比較 (2501 JP)	40
3.12	実データと生成された注文の分布の比較 (4005 JP)	41
3.13	実データと生成された注文の分布の比較 (7752 JP)	42
3.14	実データと生成された注文の分布の比較 (7911 JP)	43
3.15	学習における各種指標の遷移	44
4.1	GAN Evaluator の概要	50
4.2	PGGAN の概略 (簡易版)	51
4.3	対数リターンの Autocorrelation Function (ACF)	59
4.4	対数リターンの Autocorrelation Function (ACF) Case #0	60
4.5	対数リターンの Autocorrelation Function (ACF) Case #1	61
4.6	対数リターンの Autocorrelation Function (ACF) Case #2	61
4.7	対数リターンの Autocorrelation Function (ACF) Case #3	62
4.8	対数リターンの Autocorrelation Function (ACF) Case #4	62
4.9	対数リターンの Autocorrelation Function (ACF) Case #5	63
4.10	対数リターンの Autocorrelation Function (ACF) Case #6	63
4.11	対数リターンの Autocorrelation Function (ACF) Case #7	64

4.12 対数リターンの Autocorrelation Function (ACF) Case #9	64
4.13 対数リターンの Autocorrelation Function (ACF) Case #10	65
4.14 対数リターンの Autocorrelation Function (ACF) Case #11	65
4.15 絶対対数リターンのプロット	67
4.16 絶対対数リターンのプロット (Case #0)	68
4.17 絶対対数リターンのプロット (Case #1)	69
4.18 絶対対数リターンのプロット (Case #2)	70
4.19 絶対対数リターンのプロット (Case #3)	71
4.20 絶対対数リターンのプロット (Case #4)	72
4.21 絶対対数リターンのプロット (Case #5)	73
4.22 絶対対数リターンのプロット (Case #6)	74
4.23 絶対対数リターンのプロット (Case #7)	75
4.24 絶対対数リターンのプロット (Case #9)	76
4.25 絶対対数リターンのプロット (Case #10)	77
4.26 絶対対数リターンのプロット (Case #11)	78
4.27 Realized Volatility の Autocorrelation Function (ACF)	79
4.28 Realized Volatility の ACF (Case #0)	80
4.29 Realized Volatility の ACF (Case #1)	81
4.30 Realized Volatility の ACF (Case #2)	81
4.31 Realized Volatility の ACF (Case #3)	82
4.32 Realized Volatility の ACF (Case #4)	82
4.33 Realized Volatility の ACF (Case #5)	83
4.34 Realized Volatility の ACF (Case #6)	83
4.35 Realized Volatility の ACF (Case #7)	84
4.36 Realized Volatility の ACF (Case #9)	84
4.37 Realized Volatility の ACF (Case #10)	85
4.38 Realized Volatility の ACF (Case #11)	85
4.39 対数リターンの異なるラグごとの尖度のプロット	86
4.40 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #0)	87
4.41 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #1)	87
4.42 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #2)	88
4.43 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #3)	88
4.44 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #4)	89
4.45 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #5)	89
4.46 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #6)	90
4.47 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #7)	90
4.48 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #9)	91
4.49 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #10)	91

4.50 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #11)	92
5.1 人工市場データマイニングプラットフォームの概要	97
5.2 人工市場モデルの概要	99
5.3 アテンションベースのモデルの結果	104
5.4 CNN ベースのモデルの結果	105
5.5 LSTM ベースのモデルの結果	106
6.1 本章における、人工市場データマイニングプラットフォームの詳細	112
6.2 すべてのパラメータセットの結果のグラフ	117
7.1 パラメータチューニングのスキーム	123
7.2 カスタム DDPG のモデル概要	126
7.3 カスタム SAC のモデル概要	127
7.4 FNNA のイメージ	130
7.5 Seed Fixer のイメージ	131
7.6 ベースラインモデル (TPE: Optuna) の損失関数の推移	137
7.7 DDPG + AC + FNNA + SF の損失関数の推移	137
7.8 DDPG + AC + SF の損失関数の推移	138
7.9 DDPG + AC + FNNA の損失関数の推移	138
7.10 DDPG + AC の損失関数の推移	139
7.11 SAC + FNNA + SF の損失関数の推移	139
7.12 SAC + SF の損失関数の推移	140
7.13 SAC + FNNA の損失関数の推移	140
7.14 SAC の損失関数の推移	141
8.1 Actor-Critic によるパラメータチューニングにおける GAN Evaluator の挿入 .	145
8.2 実際の実装の概要	145
8.3 最高パラメータに対する評価値の変化 (学習時)	151
8.4 各 epoch で探索されたパラメータに対する評価値の平均値の変化	151
8.5 各 epoch までの評価値が最高であったパラメータセットの n_{agent} (青) および各 epoch で探索された n_{agent} の平均値(オレンジ)の変化	152
8.6 各 epoch までの評価値が最高であったパラメータセットの w_F (青) および各 epoch で探索された w_F の平均値(オレンジ)の変化	153
8.7 各 epoch までの評価値が最高であったパラメータセットの w_C (青) および各 epoch で探索された w_C の平均値(オレンジ)の変化	153
8.8 各 epoch までの評価値が最高であったパラメータセットの σ (青) および各 epoch で探索された σ の平均値(オレンジ)の変化	154
8.9 各 epoch までの評価値が最高であったパラメータセットの τ_{\min}^* (青) および各 epoch で探索された τ_{\min}^* の平均値(オレンジ)の変化	154

8.10 各 epoch までの評価値が最高であったパラメータセットの τ_{\max}^* (青) および 各 epoch で探索された τ_{\max}^* の平均値(オレンジ)の変化	155
8.11 各 epoch までの評価値が最高であったパラメータセットの τ_{\min} (青) および 各 epoch で探索された τ_{\min} の平均値(オレンジ)の変化	155
8.12 各 epoch までの評価値が最高であったパラメータセットの τ_{\max} (青) および 各 epoch で探索された τ_{\max} の平均値(オレンジ)の変化	156
8.13 各 epoch までの評価値が最高であったパラメータセットの k_{\min} (青) および 各 epoch で探索された k_{\min} の平均値(オレンジ)の変化	157
8.14 各 epoch までの評価値が最高であったパラメータセットの k_{\max} (青) および 各 epoch で探索された k_{\max} の平均値(オレンジ)の変化	157

表一覧

3.1	選ばれた 10 銘柄の概要	28
3.2	すべての結果一覧 (KLD)	31
3.3	すべての結果一覧 (MSE)	32
4.1	アブレーションテストの結果	58
6.1	AggOG と ABOG の NLL の評価値の一覧	116
7.1	モデルと追加要素の組み合わせパターン	124
7.2	全モデル一覧	125
7.3	評価結果 (10 試行の統計値)	135

第I部

初めに

第1章 研究概要

1.1 研究背景と目的

金融市場において、マルチエージェントシミュレーションとデータマイニングはとても有用な技術である。マルチエージェントシミュレーションとは、コンピューター上に仮想の世界と仮想のエージェントを作成し、エージェントのミクロな行動の積み上げとして世界全体の動きを再現する手法であり、複雑系の分析に有用である。一方で、データマイニングは、世の中に存在するデータを用いて、様々な知識や現象をとらえようという取り組みである。特に、金融市場において、マルチエージェントシミュレーションは、実市場で検証できないシナリオの検証など[1, 2]で有用であり、データマイニングは、利益を追求するなどの場面で多く使われてきている[3, 4]。

マルチエージェントシミュレーションは、図1.1で示している通り、未知のシチュエーションにも対応可能であるというメリットと現象理解のツールとして有用であるというメリットが存在する。一方で、出力データに現実味がないことや、シミュレーションの良さや妥当性の評価基準があいまいであるというデメリットも存在する。加えて、マルチエージェントシミュレーションは、エージェントの行動を再現するために、人間が実際の行動を考えながら、重要なポイントを考慮に入れてモデリングを行うことによって解釈性や妥当性を維持している。たとえば、Axelrod[5]は“Keep It Simple Stupid”(KISS)原理を提唱し、シミュレーションは、より少ない要素を以ってより複雑な現象を再現することで現象理解に有効であると主張した。一方で、KISS原理では、複雑な現象の再現に限界があることから、Edmondsら[6]は“Keep it Descriptive Stupid”(KIDS)原理を提唱し、説明性を維持しつつ複雑な現象を再現することでシミュレーションを活用できると主張した。ただし、いずれの原理にしたがうとしても、シミュレーションモデルの作成は人間のセンスに大きく依存したものとなっている。

一方で、データマイニングのメリットとデメリットも存在する。データマイニングは、近年、非常に膨大なデータを活用できるようになったことに加えて、データマイニング技術、特にニューラルネットワークの技術向上によって、とても強力なツールとなってきている。一方で、データにとても強く依存しており、データサンプルに存在しないようなデータや現象に対して脆弱である。この対処のためには膨大なデータを用いた事前学習を行わせるというような取り組みが行われている。例えば、画像認識タスクでは、画像を分類タスクにかけて作成した学習済みモデルが様々なアーキテクチャで提案されている[7, 8, 9, 10, 11, 12]。他にも、自然言語においては、複数の事前学習専用のタスクを解か

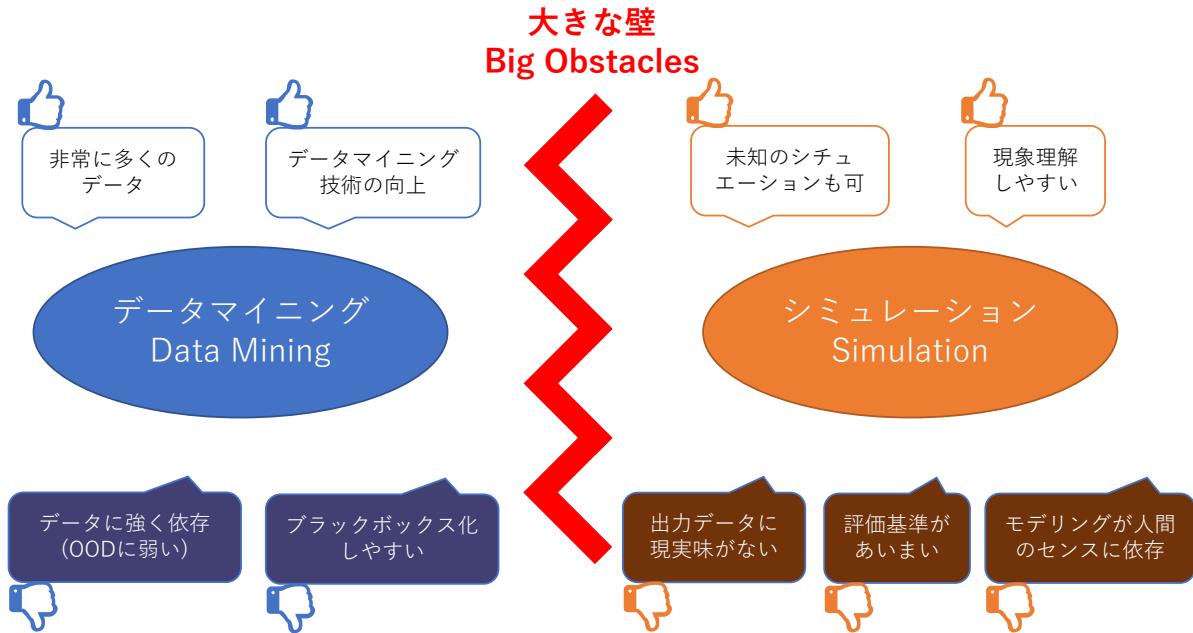


図 1.1: シミュレーションとデータマイニングのメリット・デメリット

ることで学習を行う大規模汎用言語モデルの構築 [13, 14, 15, 16, 17, 18, 19, 20, 21] が盛んにおこなわれており、金融向けのモデルも構築されている [22]. しかしながら、汎用性を高めるために、個々のタスクやデータに対してすべて事前学習モデルを構築することは困難であり、加えて、金融市場の場合は、非定常性が存在するために、モデルの構築ができたとしてもその性能は保証されない. 加えて、最新のデータマイニングで一般に使われるニューラルネットワークは、ブラックボックス化してしまうという問題点も指摘されている. ブラックボックス化してしまう問題点に対しては、解釈性を与える手法が提案されてはいる [23, 24, 25] が、まだまだ解決しきれてはいない.

これらのシミュレーションとデータマイニングのメリットデメリットは相補的な関係になっていると考える(図 1.2). たとえば、データマイニングはサンプルに存在しないデータに弱い一方で、シミュレーションは未知のシチュエーションに対応できるという強みを持つ. また、データマイニングはブラックボックス化しやすい一方で、現象理解しやすいという強みを持つ. 一方で、シミュレーションはモデリングが人間のセンスに依存しやすい一方で、データマイニングはデータという客観的なものに基づいてモデルを構築する. また、シミュレーションは出力データに現実味がないとされる一方で、データマイニングは現実の非常に多くのデータを使う. これらは、相反する特徴としてとらえられてきた結果、シミュレーションとデータマイニングの融合はなかなか行われてこなかった. しかしながら、相反すると考えるのではなく、補完的な存在であると考えることによって、その双方の強みを活用したソリューションを提案できるのではないだろうか.

すでに、筆者らの先行研究 [26, 27] では、その取り組みとして、実データに基づいてエー

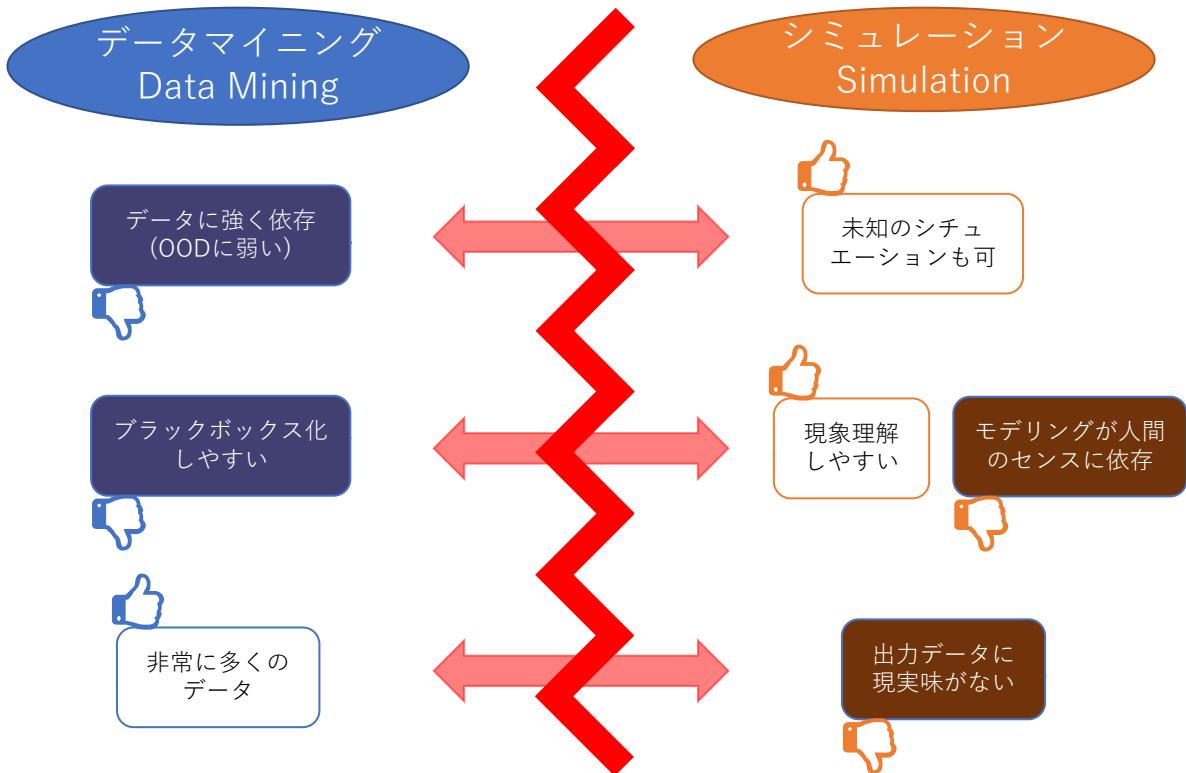


図 1.2: シミュレーションとデータマイニングのメリット・デメリットの相補的関係

ジエントを構築するという取り組みを行っている。この取り組みはマルチエージェントシミュレーションとデータマイニングの融合という目標に対して合致しているアプローチではあるものの、その双方のデメリットを取り込んだ形となってしまった。実データに基づいてエージェントの行動を生成するニューラルネットワークを用いたために、エージェントの行動をブラックボックスにしてしまい、シミュレーションの解釈性を損なった。加えて、学習データに含まれない未知のシチュエーションに脆弱になってしまった。これらにより、データマイニングの良さもシミュレーションの良さも損なった手法となってしまった。

そこで、本研究においては、マルチエージェントシミュレーションとデータマイニングを融合させ、それら双方の強みを生かす手法の構築を目指す。

1.2 論文の構成と概要

図 1.3 に、本研究の概要図を示した。3 章では、金融市場にフォーカスし、金融市場のデータを用いて、GAN を作成する。その後、その作成した GAN を用いて、シミュレーションの定量評価手法を 4 章で提案する。そして、5 章では、人工市場データマイニングプラットフォームという、人工市場内でデータマイニングの性能評価を行うスキームを

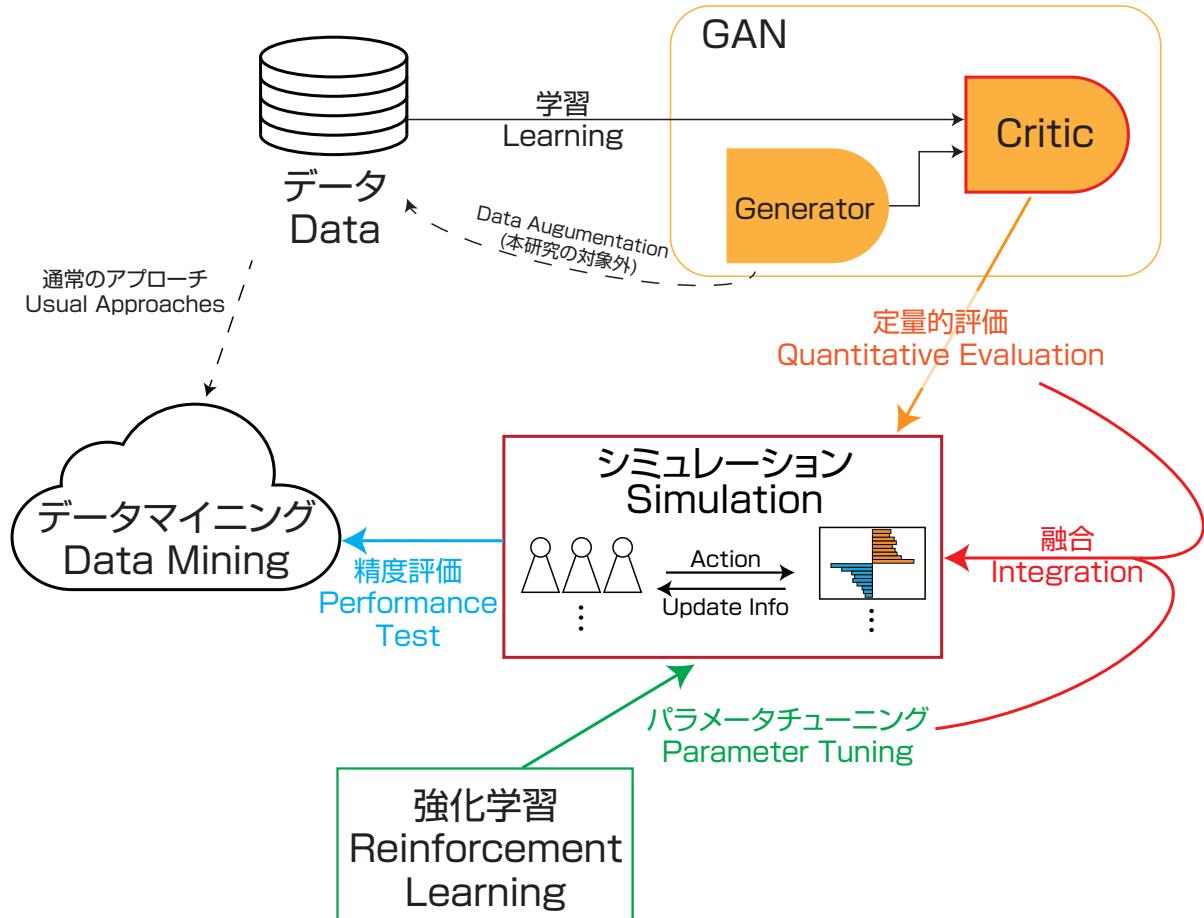


図 1.3: 研究概要

提案し、その実例として、6章では、金融市場における注文の同時性がデータマイニング手法に与える影響について検証する。また、7章では、強化学習手法を活用した、シミュレーションのパラメータチューニング手法について提案する。そして、最後に、4章のシミュレーションの定量評価手法と7章の強化学習手法を活用した、シミュレーションのパラメータチューニング手法を融合させた手法を提案・検証する。

また、2章では、関連研究をまとめ、最後の9章と10章で全体をまとめる。

なお、本博士論文には、既発表の複数の論文 ([28, 29, 30, 31, 32, 33, 34, 35]) を部分的に含んでいる。

第2章 関連研究

2.1 社会シミュレーション・マルチエージェントシミュレーションに関する研究

2.1.1 社会シミュレーションの発展と実例

マルチエージェントシミュレーションは1970年前後から多くの社会現象の解析・理解に活用されてきている。Schelling [36, 36] は、マルチエージェントシミュレーションを用いて、分居のシミュレーションを行い、人種ごとの分居が起こるメカニズムを明らかにし、のちにノーベル経済学賞を受賞した。Axelrod [37, 38] は、マルチエージェント環境において、囚人のジレンマコンテストを実施し、様々な種のエージェントを分析した。Epstein ら [39] は、アリと食料を模した人工的な世界をシミュレーションすることで、人工社会の構築可能性を示した。Lux ら [40] は、金融市場において観測される特有の現象は、エージェント間のインタラクションがなければ再現することができない、ということを示し、マルチエージェントシミュレーションの必要性を説いた。他にも様々なシミュレーションが社会的に活用されている。Sajjad ら [41] は、人口動態のシミュレーションを実データに基づいて構築した。Nonaka ら [42] は、人流シミュレーションを構築し、避難シミュレーションを実施した。Braun-Munzinger ら [43] は、債券市場向けのマルチエージェントシミュレーションを構築した。Kurahashi ら [44] は、マルチエージェントシミュレーションを用いて、COVID-19 の感染拡大防止策の検討を行った。

社会現象においては、その支配方程式が存在していない、あるいは解明されていないために、その解析が難しい。そこで、シミュレーションは、社会科学分野での有用であると主張されている [45]。また、特にマルチエージェントシミュレーションは重要であると主張する研究も存在する [46, 47]。

このように多くの活用が行われている一方で、マルチエージェントシミュレーションの構築論に関しても議論が行われている。Axelrod [5] は、“Keep It Simple Stupid”(KISS) 原理をマルチエージェントシミュレーションにおいて提唱しており、より簡単なモデルで複雑な現象を再現することにより、その本質にある現象の理解に寄与することができるとした。さらにEdmonds ら [6] は “Keep it Descriptive Stupid”(KIDS) 原理を提唱し、その目的のためであれば、KISS でなくとも、説明可能性があればよいと提唱した。寺野 [48] も同様に、KISS 原理を超えるようなエージェントシミュレーションの手法の可能性を議論している。

2.1.2 人工市場シミュレーションに関する研究

人工市場シミュレーションとは、金融市場を模したマルチエージェントシミュレーションである。ここでは、金融市場シミュレーション一般について述べる。

まず、社会シミュレーションの重要性と同様に、金融市場におけるシミュレーションの重要性は多く議論されてきている[46, 47]。前述の通り、Luxら[40]は金融市場シミュレーションにおけるエージェントインタラクションの必要性を示しており、特にマルチエージェントシミュレーションの必要性が明らかになっている。さらに、Mizuta[49]は、金融におけるマルチエージェントシミュレーションが金融の規制や制度設計に貢献できる可能性を論じている。

既存の金融理論の限界が主要な実務者から主張されていることも、人工市場シミュレーションの研究に拍車をかけている。2007年から2008年にかけて発生した金融危機(日本では通称リーマン・ショック)では、米国住宅市場の悪化による住宅ローンのデフォルト問題を契機として、投資銀行の破綻を引き起こし、さらにそれが世界的な金融市場全体の混乱を引き起こす結果となった。当時、欧州中央銀行(ECB)の総裁であったTrichetは、従来の金融理論では金融危機中の政策決定の役に立たなかつたと述べ、行動経済学やマルチエージェントシミュレーションの必要性を述べた[50]。また、投資銀行やヘッジファンドでリスク管理をとりあつかい、米国財務省でも勤務経験のあるBookstaberは、著書[51]内で金融危機を振り返って、従来の経済学では、危機の時のような歪みが増幅した状態を取り扱うことが難しく、エージェントシミュレーションのような複雑さを取り込むことのできる手法へのパラダイムシフトを推奨している。

具体的な人工市場を用いたシミュレーションは様々である。Cuiら[52]はエージェントの行動において知能がない場合(Zero-Intelligence)，一部の金融市場で見られる現象を再現できないことを示した。Toriiら[53]は価格ショックが他の株式にも伝搬するシミュレーションを実施し、そのメカニズムを分析した。Mizutaら[1]は、株式市場における価格の呼値の影響を人工市場で分析し、呼値の引き下げが市場シェアの維持には必要であると主張し、東京証券取引所における呼値の切り下げの議論に貢献した。Hiranoら[2]は、自己資本比率規制の影響を人工市場を用いて分析を行い、市場の安定性を確保するために導入されたはずの自己資本比率規制が逆に価格ショックの増幅や値上がりを抑えてしまう可能性があることを示した。他にもフラッシュクラッシュを人工市場で再現した研究なども存在する[54, 55]。

これらの人工市場シミュレーションを実現するためのプラットフォームも複数提案されている。Toriiら[56]は“Platform for large-scale and high-frequency artificial market”(Plham)[57]を提案・公開している。さらに、後続のプラットフォームとして、Java版のPlhamJ[58]やpython版のPams[59]なども提案・公開されている。また他にも、UMART[60]、Santa Fe Artificial Stock Market[61]、Agent-based Interactive Discrete Event Simulation(ABIDES)[62]なども提案されている。本研究ではPlhamJ[58]を使用している。

2.1.3 シミュレーションの評価とパラメーター管理に関する研究

マルチエージェントシミュレーションにおいては、非常に多くのモデルパラメータが採用される傾向にある。加えて、特定の社会現象の再現に当たっては、そのパラメータの調整が不可欠である。このパラメータの調整の過程の面倒さから、様々なツールが提案されている。その一つとして、Muraseら[63]が提案・公開しているOrganizing Assistants for Comprehensive and Interactive Simulations (OACIS)がある。これは、様々なパラメータの組み合わせをグリッドサーチ的に実験を行うことを管理するツールである。また、Muraseら[64]はさらに、CARAVAN (A Framework for Comprehensive Simulations of Massive Parallel Machines) という、目的変数を設定すれば、パラメータのチューニングを自動的に行えるツールも提案している。他にも、パラメータチューニングの手法としては、ベイズ推定ベースの手法が機械学習向けに提案されている。Ozakiら[65]はTree-structured Parzen Estimator (TPE) というベイズ推定ベースの手法を提案しており、Optuna[66]という形で公開されている。

2.1.4 サロゲートモデルに関する研究

サロゲートモデルとは、日本語で代理モデルの意味であり、複雑な計算を要するアルゴリズムの入出力関係を別のモデルで代理するモデルのことである。このサロゲートモデルは、古くから近似関数による計算の簡略化として用いられてきている[67]。機械学習や深層学習の発展により、複雑な入出力関係の近似が容易になったことからその手法がさらに発展してきた[68, 69, 70, 71]。

サロゲートモデルはマルチエージェントシミュレーションにおいても活用されている。Yamashitaら[72]は、マルチエージェントシミュレーションの一部をニューラルネットワークに代理することによって、人流シミュレーションにおける最適施策の探索の計算コストを削減した。Angioneら[73]は、マルチエージェントシミュレーションにおけるサロゲートモデルの活用可能性について主張している。マルチエージェントシミュレーションは、比較的計算コストの高いものが多いため、サロゲートモデルによる計算コスト削減は有用である可能性が高い。

2.2 データマイニング・深層学習に関する研究

近年、データマイニングおよび深層学習は様々な研究が行われている。ここでは、本研究に関連するテーマに絞って、先行研究を述べる。

2.2.1 金融データマイニングに関する研究

金融において、データマイニングや深層学習を活用する取り組みは非常に多い。まず、株式価格やインデックスをサポートベクターマシン(SVM)を用いて予測しようという取り組みが複数行われている[74, 75]。さらに、深層学習を用いて、市場価格やトレンド、変化点を予測しようという研究もおこなわれている[76, 77, 78, 79]。特に、Wangら[3]はCLVSA(A Convolutional LSTM Based Variational Sequence-to-Sequence Model with Attention for Predicting Trends of Financial Markets)という、価格トレンド予測のために、Long Short Term Memory(LSTM)やSequence-to-sequenceなどのモデルを組み合わせたモデルを提案している。Tashiroら[80]は、高頻度なミリ秒レベルの時間スケールでの価格予測モデルを提案した。彼らは、さらに、LSTMの代わりにCNNを用いた手法も提案している[4]おり、リカレントな構造を持つLSTMよりもCNNの方が計算の高速化が期待できるという点で、ミリ秒レベル予測にも有効であると考えられる。

他にも、金融市場における特異的なパターンを見つけ出そうとする研究も存在する。Nanex[81]は、注文データの中で、特に特徴的な繰り返しを行うトレーディング行動などの注文行動を抽出し、報告している。Cont[82]は、Stylized Factsという、金融市場で観測される特有の現象を統計的分析から明らかにした。Miyazakiら[83]は、実際の注文の中から違法な注文をガウス混合モデルを用いて検出する手法を提案した。Hiranoら[84]はStochastic Trading Behavior Model(STBM)というLSTMベースのモデルを提案し、実際の注文データを用いて、実際のトレーダーの注文行動を予測するモデルを構築した。さらに、Hiranoら[85]は、Transformers[13]やResidual Blocks[9]を活用した発展的なモデルも提案しており、後続の研究[26, 27]で、このモデルをマルチエージェントシミュレーションのエージェントとして活用する手法を提案・実践している。

2.2.2 敵対的生成ネットワーク(GAN)

GANはGoodfellowら[86]により、敵対的にGeneratorとDiscriminatorが学習する機構として提案された。Mirzaら[87]は、条件付きの学習機構を導入したConditional GANを提案した。Radfordら[88]は、CNNベースのDeep Convolutional GAN(DCGAN)を画像生成向けに提案した。GANの学習を安定化させるために損失関数にMean Square Error(MSE)を用いたLeast-Squares GAN[89]や、f-divergenceに基づいたGeneralized f-GAN[90]、Laplacian Pyramidを活用したLaplacian Pyramid GAN[91]、VAEを取り込んだVariational Autoencoder GAN[92]、pix2pixとしても知られるImage-to-image Translation GAN[93]、Attention機構をSelf-attention GAN[94]、画像のスタイル変更のためのCycle GAN[95]やStyle GAN[96]、より画質の良い画像生成を目指すProgressive Growing GAN[97]などが提案されている。加えて、GANの拡張として、画像の埋め込みベクトル作成のために敵対的な機構を活用するAdversarial Feature Learning[98]や、アノマリーの検知のためにGANを活用する手法[99, 100, 101]などが提案されている。さら

なるGANの学習における安定性に関する数理的解析の議論[102]から, Wasserstein距離に基づいたWassersteinGAN(WGAN)[103]が提案されている。このWGANにおける学習の安定性のための1-Lipschitz制約を深層学習に取り込む手法として, Gradient Penalty[104]とSpectral Normalization[105]も提案されている。

金融市場向けのGANとしては, 注文時系列を生成するGANが主に研究されている。その代表としては, StockGAN[106]があげられる。StockGANでは, より現実的な注文時系列のために, 二重連続オークションを再現するためのContinuous Double Auction(CDA)Networkを採用することで, 最良気配値の更新も可能にするGANを提案している。Naritomiら[107]は金融市場向けに基本的なGANを構築し, そのGANによるデータ拡張が金融市場の株価予測に有効であることを示した。Colettaら[108]はCGANベースの金融向けGANを構築し, マーケットインパクトなどのタスクでの実用を行った。加えて, GANの機構を金融のタスクに活用した研究もおこなわれている[109, 94]。

2.2.3 強化学習の関連研究

強化学習はとても多く研究されている。Q-learningは有名なOff-policyの強化学習であり, Q-tableの学習を通じて戦略を学習する手法である[110]。学習理論という観点では, Temporal Differenceというベルマン方程式に基づいたQ値の更新手法が広く使われている[111]。Tesauroは, バックギャモンにこのTemporal Differenceに基づいた学習を実施することで, ゲームへの応用を実践した[112]。単純な強化学習の手法として, On-policyのSarsa(State–Action–Reward–State–Action)も提案されている[113]。他にも多くの強化学習手法が提案されている。最も大きな手法における変化は, 深層学習の導入である。深層学習の発展後, 強化学習においても深層学習の活用が進んだ。Mnihら[114]は, Deep Q-learning(Deep Q-Network, DQN)を提案し, Atari Learning Environment(ALE)[115]で, 人間よりも高い性能を達成した。くわえて, DQNの拡張として, 学習の安定性を維持するためにTarget Networkを活用するDouble DQN(DDQN)[116]が提案されている。さらに学習の性能を高めるために, 新たな価値関数を導入したDueling DDQN[117]も提案されている。Q-learningの拡張としては, Asynchronous Advantage Actor-Critic(A3C)[118]が非同期的な深層強化学習手法として提案されている。のちに, 非同期性を取り除いたActor-Critic[119]やAdvantage Actor-Critic(A2C)[120]が提案されている。HesselはRainBow[121]という上記の様々な手法を組み合わせた手法を提案している。他にも, Prioritized Experience Replayを活用したApe-X DQN[122]や, LSTM[123]を活用したR2D2[124]なども発展的なモデルとして提案されている。Silverらは自己対戦を学習に組み込むことでチェスや将棋や囲碁で高い性能を達成した[125]。この手法は, “Alpha Go Zero”[126]として知られている。また, 本研究に関連する強化学習手法としては, 連続値を扱うことのできるDeep Deterministic Policy Gradient(DDPG)[127]や, 方策エントロピーを導入することで探索力を向上させたSoft Actor-Critic(SAC)[128, 129]なども存在する。

2.3 人工市場シミュレーションと金融データマイニングの融合に関する研究

まず、金融市場において、シミュレーションとデータマイニングを融合した研究としては、Maeda らの研究があげられる。Maeda ら [130] は、人工市場シミュレーションを用いて、深層強化学習を学習させるた。また、オプション市場においては、Deep Hedging [131] という、シミュレートされた価格時系列上でオプションヘッジの戦略を学習させる手法が提案されている。また、Hirano ら [26, 27] は、データマイニングで構築したエージェントモデルをシミュレーションで活用する手法を提案している。

第II部

GANの構築とシミュレーションの評価

第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配GAN

3.1 本章における問題意識と背景

金融市場において、実現された価格時系列や注文時系列は、起こりうる多様な時系列のうちの一つでしかない。そもそも、金融市場の時系列変化は決定論的なものではなく、事前に一意に定まらないものである。その多様な可能性の中で、実現された時系列はほんのわずか一部である。別の言い方をすれば、金融市場には非定常性があるともいえる。つまり、金融市場をモデル化する場合には、過去の時系列はあくまで実現した時系列というバイアスが存在しており、過去の時系列に現れないような現象が将来発生する可能性を考慮に入れることが難しいことを意味する。

1つ1つの注文のバリエーションには限りがある。つまり、売り注文か買い注文なのか、成行注文なのか指値注文なのか、さらには、いくらでどの数量の注文なのかという組み合わせは、常識的な範囲では限られている。しかしながら、この注文を時系列方向に積み上げていくと、その状態空間の可能性は膨大な大きさとなる。くわえて、金融市場においては、定常性が存在しておらず、同じパターンで時系列が現れるとは言えない。そのためには、過去に実現した時系列だけを学習するだけでは学習としては不十分であることが多い。

こういった、データの不十分性や非定常性などに対応するためには、モデルの学習に活用するデータ量を増やすデータ拡張が有効な手法である。例えば、取引戦略の学習を行うとしても、データが多い方が性能が向上することが期待できるほか、バックテストを行う際にデータ拡張を行うことで、より正確な性能評価を行うことができるであろう。加えて、時系列データの拡張を行うことにより、ポートフォリオの価値やリスク評価等もより正確に行えるであろうことが期待できる。

このデータ拡張という観点では、非常に多くの研究が行われてきている。データ拡張という目的では、大きく分けると二つのアプローチが想定できる。一つ目が人工市場シミュレーションによるデータ拡張、もう一つがGANによるデータ拡張である。本章では、GANによるデータ拡張について述べるが、本論文内では人工市場を取り扱うものの、データ拡張として人工市場を用いることはないので、データ拡張としての人工市場についてもここで述べる。

14第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配GAN

人工市場シミュレーションは、仮想の状況における、コンピューター上の市場のシミュレーションを行う手法であるため、現実に存在しない状況や設定も実験することが可能である。加えて、新しい規制の導入などの外部要因に関しても、未知であろうとも実験が可能であることが強みである。そのため、人工市場における設定を変更して実験を行うことで、有用な示唆を得ることができる。一般に、人工市場シミュレーションはデータ拡張手法として考えられることは少ないものの、従来のデータに依存することなく、有り得るが実際には起きていない状況に関するデータを拡張できるという点で、有望なデータ拡張として考えることも可能であると考えている。

もう一方のアプローチはGANであり、金融市場においては、過去のデータに基づき、時系列データなどを新たに生成する手法である。無論、VAEなどの生成ネットワークを活用すれば、データ拡張を行うことは可能であるが、現時点では金融時系列の拡張として最も一般的な手法がGANであることから、生成ネットワークの代表として、GANを取り上げている。このGANによるデータ拡張は、人工市場よりも現実的なアプローチであり、より現実的な時系列を生成する傾向にある。一方で、GANの構築に当たっては、データの背景にある分布構造を暗黙的に学習するために、非常に多くのデータが必要になるという問題点は存在する。

本章においては、この金融向けGANの構築を行う。2.2.2節ですでに先行研究を述べた通り、金融向けのGANの先行研究はいくつか存在する。Stock-GAN (S-GAN) [106] や Market GAN [107] などが提案されている。しかしながら、これらのGANは金融市場の注文生成という点において、現実的ではない部分が存在する。GANの学習にあたっては、一般的に、GeneratorとCritic (Discriminator) の間での勾配の接続が不可欠である。これは、Generatorが生成モデルとして学習を行うにあたって、勾配をCriticを通じて得る必要があるためである。この勾配の接続の必要性が生成される注文時系列を非現実的にする原因となっている。例えば、S-GAN [106]の場合、勾配接続可能なGeneratorによる生成を行うために、売り買い、注文タイプ、価格、数量を連続値として生成してしまっている。実際の市場での注文を考えると、売り買いのシグナルが0から1の確率値とすることはできず、注文時には、売りか買いかのどちらかにしなければならない。加えて、注文タイプも同様に成行なのか、指値注文なのかも注文時に決めている必要があり、こちらも0から1の確率値として生成するのは若干不適切である。さらに、価格と数量については、実際の市場においては、呼値やロットサイズといった、最小の単位が定められているため、連続値をそういった離散の値に変換をする必要が出てきてしまう。このような勾配接続のための出力の連続化は、Generatorの出力からその結果を丸め込む必要があることから、生成性能の低下を引き起こしかねない上、純粋な生成機としては人間でも容易に偽物であると判別できてしまう注文を生成してしまうという問題点が存在する。一方で、Market GAN [107]は、クラスの確率として生成を行うため、上記の問題が緩和されている。しかしながら、S-GANにも存在した、売買や注文タイプといったようなものに加えて、価格や数量までも確率値として生成し、それをCriticに判定させるのもまた、同様の問題を孕んでいる。つまり、理想的に言えば、Criticは、実際の市場が受付可能な形式で入力を受

け取り、それに対して、Fake なのか Real なのかという判定を行うのが理想であり、それが達成されていないということである。無論、S-GAN も Market GAN も生成された注文を丸めたり、確率値に基づいてサンプリングを行えば、実際の市場が受付可能な注文形式に落とし込むことは可能であるものの、そのサンプリングや丸め込みのアルゴリズムが Generator と Critic の間に挟まっていることが問題である。これにより、Critic は、本来判定すべき点に着目できていないのではないだろうか。例えば、売買のうち、買い注文である確率が 0.6 であるという生成が行われたとして、それが Critic に入力されたとする。このような注文は、明らかに Fake の注文である。なぜならば、実際の実現した注文は、売りか買いかのどちらかの確率が 1.0 であり、他方が 0.0 であるからだ。この問題は S-GAN と Market GAN の双方で起こる問題である。さらに、S-GAN だけで起こりうる問題として、100.0123 円の買い注文、というような注文が生成されうるということである。国や市場によってその単位はまちまちであるものの、呼値を考慮するとこのような注文は不可能である。東京証券取引所の場合、少なくとも 0.1 円より大きい呼値が設定されているので、100.0 円または 100.1 円のような離散的な価格設定しかできないはずである。

ここまで議論したように、生成される注文は、実際の市場でも受付可能な注文であるべきであると考える。もちろん、売り買いのシグナルを確率的に生成するということ自体は、人間も同様にどちらで注文を出すか迷うのと同様に問題ないと考えられる。そのため、単に確率値で生成して、それをそのまま Critic に入力するのではなく、実際に、市場が受付可能な注文形式にサンプリングするなどしてから Critic に入力すべきであると考えている。

加えて、買いと売りの注文の状態空間が連続的に接続していることも問題があると考えている。例えば、売りと買いの注文確率を $[0, 1]$ で表現して、そのまま Critic に入力することを考える場合、買いと売りの注文の状態空間が連続的に接続されていることになる。例えば、200 円で 100 株の注文を出すことを考える。この時、最良気配値が 190 円付近であったとする。この時に、この注文が買い注文であったとすると、これは、裁量気配値を上回る、実質的に成行注文と大差ないテイク注文ということになる。一方で、この注文が売り注文であったとすると、これは、すぐには執行されず、板に残るメイク注文であることになる。たった売りと買いの方向が違うだけにも関わらず、注文の意味合いは全く異なる。例えば、売りと買いの注文が 50% ずつであるとすると、そもそもメイク注文なのかテイク注文のかさえも五分五分の状況であり、これを Critic の入力として用いるのは Critic にとって判定困難であると考えられる。特に、メイク注文は自然でもテイク注文が明らかにおかしいシチュエーションだった時の Critic の Fake/Real の判定は混迷を極めるであろう。そのため、Critic にはサンプリングを行なって、どちらなのかを明確にしたのちに入力されることが必要であると考える。

これらの議論に基づくと、注文の特徴量の離散性を GAN の機構に考慮を入れた注文生成が行われるべきであると考える。

この問題解決にあたって、最も大切であるのが、GAN の学習における Generator と Critic の勾配接続である。もし、この勾配接続が不要であったとすると、Generator はサンプリ

ングなどのプロセスを導入することが可能になるため、上記で議論したような現実的でない注文を生成してしまう問題や、確率のまま Critic に入力するなどの問題が解決することができる。

本章では、これらの問題に対処するため、Policy Gradient Stock GAN (PGSGAN) を提案する。上記で述べた問題を解決するために、Generator と Critic の勾配接続を切る代わりに、Policy Gradient を用いることにより、Generator の学習を可能にすることを目指す。Policy Gradient とは、深層強化学習で頻繁に使われるある学習理論であり、GAN の Generator と Critic を強化学習のフレームワークに落とし込むことで、勾配接続の切れた状態であっても Generator が学習するようにすることができる。これにより、Generator と Critic の勾配接続が必要なくなり、より現実的な離散的な注文を生成することが可能になることを目指す。

3.2 提案手法 1: Policy Gradient Stock GAN (PGSGAN)

PGSGAN は、過去の注文と現在の市場状況を入力にとり、次の注文を生成する金融市場向けの GAN である。PGSGAN はオリジナルの GAN [86] をベースとしながら、様々な改良を組み込んでいる。過去の注文を Generator と Critic の双方に入力として入れるという点では、Conditional GAN [87] 的な構造となっている。また、前節で述べた通り、Policy Gradient Theorem [132, 133, 134] を学習理論に活用している。より具体的な構造としては、GAN の機構としては、Wasserstein GAN (WGAN) [103] を採用しており、Policy Gradient の実装にあたっては、ベースライン付き REINFORCE [135] を採用している。加えて、学習の安定性の確保のために、Batch Normalization [136]、Layer Normalization [137]、Spectral Normalization [105] も使用している。

具体的な実装や学習理論は、以下で順に説明する。

3.2.1 PGSGAN の構造

図 3.1 は、PGSGAN の概要を示している。

Generator は、条件として与えられる市場の過去注文データと生成のための乱数を受け取る。本研究では、過去 20 注文と現在の売りと買いの裁量気配値を条件として入力している。過去の注文時系列には、売りか買いか、新規注文か取り消し注文か、成行注文か否か、最良気配値からの距離 (Ticks)、注文数量 (最小注文単位で割ったもの) が含まれている。一方で、生成のための乱数は、128 次元の乱数が入力される。

これらの入力に基づいて、Generator は Policy を生成する。この Policy というのは、生成すべき注文の分布のようなものであり、ここからサンプリングを行うことで、偽物の注

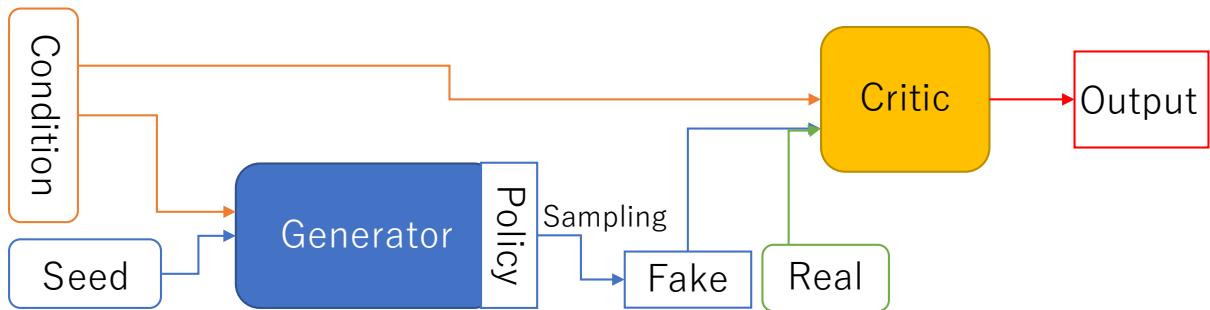


図 3.1: PGSGAN の概要. Generator は、条件として与えられる市場の過去注文データと生成のための乱数を受け取る. Generator が確率論的な Policy が生成されたのちに、重み付きルーレット選択により、偽物の注文をサンプリングする. Critic は生成された注文、もしくは、実際の注文を入力として受け取り、これを 1-Lipschitz 制約のもとでスカラー値にマップする.

文を生成する. 本研究では、Generator の構造として、14 層の畠み込み層と 5 層の線形層を用いて、以下の Policy の生成を行なっている.

- 売りか買いか？ – 2 クラス (確率値)
- 新規注文か？ – 2 クラス (確率値)
- 成行注文であるか？(Is MO) – 2 クラス (確率値)
- 相対価格 (最良気配値からの Tick 数) – 40 クラス (0 – 39までのクラスに対応する確率値)¹
- 注文数量 (最小注文単位で割った数値) – 40 クラス (0 – 39までのクラスに対応する確率値)¹

こうして、確率論的な Policy が生成されたのちに、これに基づいて Generator は重み付きルーレット選択により、偽物の注文をサンプリングする.

Critic は生成された注文、もしくは、実際の注文を入力として受け取り、これを 1-Lipschitz 制約のもとでスカラー値にマップする. Critic については、基本的な WGAN と同じである. Critic は二つの入力を受け取る. 一つが、Generator と同じ条件データである. もう一方は、Generator が生成した偽の注文か、実際の注文である. これに基づいて、入力された注文が、条件データの次の注文として正しいかどうかを判定できるようなスカラー値へのマッピングを行えるように学習を行う.

¹ 裁量気配値からの Tick 数が 40 以上のものや注文数量が 4000 以上のものについては、40 番目のクラスに含むこととする. 裁量気配値からの Tick 数が 40 以上の注文や注文数量が 4000 以上の注文は稀であるものの、存在する. しかしながら、その比率というのは今回使用した東京証券取引所のデータ内では非常に少ない. そのため、本研究では詳細にモデルに考慮する必要がないものであるとして考えている.

本章においては、実データも、偽のデータの生成と同じように、40 クラス以上離れた注文価格と注文数量は同様に 40 クラスまでに丸め込むこととしている。また、実データが成り行き注文であった場合には、価格のクラスは 0 番目に指定する。

3.2.2 PGSGAN の学習理論

偽注文の生成の過程にサンプリングを導入した結果として、Generator と Critic の勾配の接続が失われる。一般的な GAN や金融市場における上述の GAN の先行研究では、この勾配接続に依存して、Generator の学習をおこなってきていた。しかしながら、PGSGAN では、完全にこの接続を捨てているため、一般的な GAN の学習理論を適用できない。

そのため、ここでは、強化学習でよく使用される Policy Gradient を用いた新しい学習法を提案する。

以下では、下記の表記を用いる。

- z : 亂数。主に Generator に使用される。本研究では $z \in \mathbb{R}^{128}$ を採用している。
- \mathbb{P}_z : 亂数分布
- \mathbb{P}_r : 実データの分布
- $C(x)$: Critic. 関数として表記しており、入力に x をとり、出力はスカラー値である。²
- $G(z)$: Generator. 関数として表記しており、入力に乱数 z をとり、policy を出力する。Policy は注文分布として機能する。²
- $\tilde{x} \sim G(z)$: \tilde{x} は、Policy からサンプリングされた偽注文であり、生成された Policy $G(z)$ に基づく。
- θ_C, θ_G : Critic と Generator の内部パラメータ。ここでは、Critic と Generator はニューラルネットワークとして想定をして、そのパラメータと考える。
- L_C, L_G : Critic と Generator の損失関数。
- $\|f\|_{L \leq 1}$: 1-Lipschitz 制約のついた任意関数 f 。
- $p_{G(z)}(\tilde{x})$: 生成された Policy $G(z)$ に基づく場合に、サンプルされた偽注文 \tilde{x} が生成される確率。
- $NLL_{G(z)}(\tilde{x})$: 生成された Policy $G(z)$ に基づく場合に、サンプルされた偽注文 \tilde{x} が生成される確率の負の対数尤度。 $NLL_{G(z)}(\tilde{x}) = -\ln \{p_{G(z)}(\tilde{x})\}$ である。

²正確には条件付き生成を行うため、入力に条件となるヒストリカルデータも含まれるが、表記の簡便化のため、ここでは表記しない。

まず、 PGSGAN は下記の min-max ゲームとして定式化される。

$$\min_G \max_{\|C\|_{L \leq 1}} L_{\text{GAN}}(G, C). \quad (3.1)$$

ここで、

$$L_{\text{GAN}}(G, C) := \mathbb{E}_{x \sim \mathbb{P}_r} [C(x)] - \mathbb{E}_{z \sim \mathbb{P}_z} [\mathbb{E}_{\tilde{x} \sim G(z)} [C(\tilde{x})]].$$

である。この定式化は、元の WGAN と基本的に同様である。しかしながら、 $\mathbb{E}_{\tilde{x} \sim G(z)}$ の Policy からのサンプリングの部分が追加されており、この部分が本研究での肝である。

Critic の目的関数は、

$$\max_{\|C\|_{L \leq 1}} \{\mathbb{E}_{x \sim \mathbb{P}_r} [C(x)] - \mathbb{E}_{\tilde{x} \sim G(z)} [C(\tilde{x})]\} \quad (3.2)$$

と表記できる。これは、Generator およびその入力乱数は Critic には直接関係なく、生成された偽の注文だけが Critic に影響するため、単純な目的関数最大化の問題として定式化できる。加えて、WGAN は Critic の偽注文と本物の注文に対する出力のスカラー値の差を 1-Lipschitz 制約下で最大化する問題と定式化されるために上記の式で表記できる。これらを踏まえると、Critic の損失関数は以下のように表記できる。

$$L_C := \mathbb{E}_{\tilde{x} \sim G(z)} [C(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r} [C(x)]. \quad (3.3)$$

一方で、Generator の学習理論は、オリジナルの WGAN よりは複雑になる。学習の概要は図 3.2 に示している。

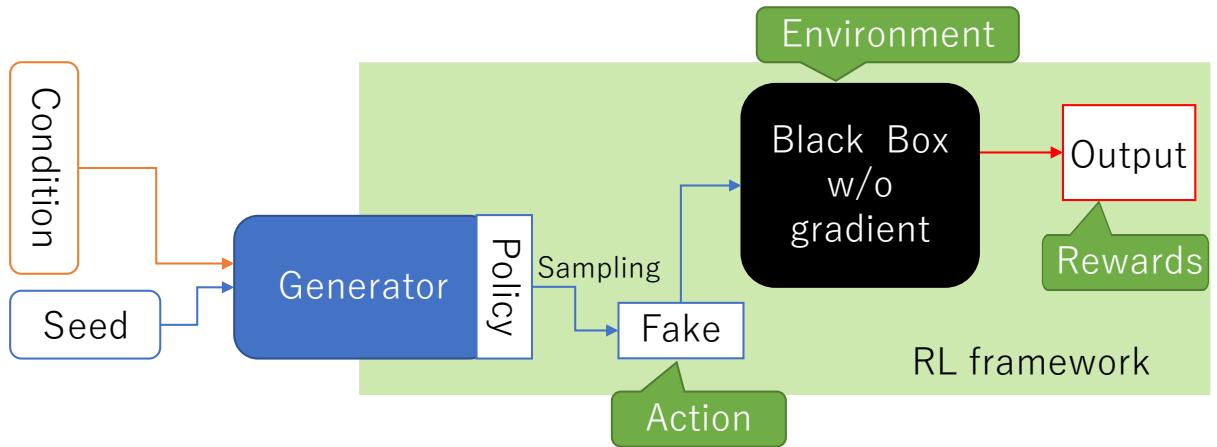


図 3.2: PGSGAN の Generator の学習を強化学習の観点から示した概要。Generator が Policy を生成し、そこからサンプリングされる偽注文は強化学習における Action としてみなすことができる。さらに、Critic は、強化学習における環境としてみなすことができ、最終的な出力が報酬としてみなすことができる。

20第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配GAN

Generator の目的関数は次の通り表される.

$$\min_G \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [C(x)] - \mathbb{E}_{z \sim \mathbb{P}_z} [\mathbb{E}_{\tilde{x} \sim G(z)} [C(\tilde{x})]] \right\}. \quad (3.4)$$

一般的な WGAN と同様に第1項は実データの Critic 評価値であるので, Generator が操作できない項になる. そのため, もっと簡便に記載することができる.

$$\max_G \mathbb{E}_{z \sim \mathbb{P}_z} [\mathbb{E}_{\tilde{x} \sim G(z)} [C(\tilde{x})]]. \quad (3.5)$$

この目的関数は, Generator の出力からのサンプリングプロセス $\mathbb{E}_{\tilde{x} \sim G(z)}$ が存在するために, Generator と Critic の勾配接続が切れており, 勾配逆伝播ができない. そのため, ここで強化学習でよく使われる Policy Gradient を導入する. 具体的な手法としては, REINFORCE [135] を使用する.

図3.2に示す通り, Generator の学習は, 強化学習の枠組みに落とし込んで考えることができる. Generator は報酬最大化を目指して, Action をコントロールする Policy を生成する. そして, Policy に基づいて, Action としてみなすことのできる偽注文が生成される. その上で, 未知の環境を通じて報酬が獲得される. ここで, 報酬は Critic の出力が対応する. そして, 最終的に, 報酬の基づいて Generator が更新される.

REINFORCE のパラメータ更新は下記の通りである.

$$\theta \leftarrow \theta + \alpha \cdot G \nabla_\theta \ln \pi_\theta(a|s). \quad (3.6)$$

ここで, θ はモデルパラメータで, α は学習率, G はリターン (通常は, 割引将来報酬の合計だが, 本研究では, Action a から得られる即時報酬とみなす), $\pi_\theta(a|s)$ は State s における, Policy π_θ に基づいた Action a を選択する確率である. さらに, ベースラインを導入すると, 式3.6は以下の通りに変換される.

$$\theta \leftarrow \theta + \alpha \cdot (G - B) \nabla_\theta \ln \pi_\theta(a|s), \quad (3.7)$$

ここで, B はベースラインである. 本研究では学習バッチごとの G の平均をベースラインとして採用している.

PGSGANにおいて, REINFORCEを適用すると, パラメーター更新は以下のようになる.

$$\theta_G \leftarrow \theta_G + \alpha(C(\tilde{x}) - B) \nabla_{\theta_G} \ln p_{G(z)}(\tilde{x}). \quad (3.8)$$

ここで, $\tilde{x} \sim G(z)$ であり, $z \sim \mathbb{P}_z$ であり, B はバッチごとの $C(\tilde{x})$ の平均である. そして, Generator の損失関数は以下の通り定義される.

$$L_G := -(C(\tilde{x}) - B) \ln p_{G(z)}(\tilde{x}) \quad (3.9)$$

$$= (C(\tilde{x}) - B) \cdot \text{NLL}_{G(z)}(\tilde{x}). \quad (3.10)$$

これらの理論に基づいて, Generator は学習可能になる.

3.2.3 PGSGAN の実際の実装

ここでは、 PGSGAN の学習理論ではカバーできない、実際の実装に関する注意点について追加で記載する。

PGSGANにおいて、すべての深層学習の層に Spectral Normalization を適用することが重要である。たとえば、Layer Normalization に対しても、Spectral Normalization を適用することで、厳密な 1-Lipschitz 制約を適用することが必要である。加えて、1-Lipschitz 制約は本来、Critic にのみ適用すれば充分であるが、学習の安定のために Generator にも適用することが良いと知られているため、本研究でも適用する。

また、条件データであるヒストリカルデータを処理するレイヤーは Critic と Generator で同じであるが、別々の学習を厳密に実現するために、重みを共有しない設定とする。

具体的な層の積み重ねについては、図 3.3, 3.4 とともに後述するが、実際の実装としては、Critic のパラメータ数が 113,071, Generator のパラメータ数が 141,625 である。なお、条件データを処理する共通の機構部の 44,750 を含んでいる。

図 3.3 に Generator の詳細を示した。基本的な機構としては、CNN を用いている。これは、Attention 層を用いる場合は、厳密な 1-Lipschitz 制約を満たすことができないために、Attention 系統のネットワークを避けたためである。加えて、条件データを処理する部分としては、CNN と Average Pooling 層を用いた。これは、時系列方向の順番はとても短い時間の間の順番は必然性がないと想定し、時系列方向の揺らぎによりロバストな機構を導入するというモチベーションで採用している。ヒストリカルデータ用のネットワークの出力と現在の裁量気配値情報を結合したのちに、Dilated Convolution [138] と循環畳み込みを用いて、それらを等価に畳み込みを行う。それらによる畳み込みを完了したのちに、線形層で Logits に変換することで、注文のそれぞれの特徴を生成する。式 3.10 で述べた通り、Generator の損失関数は下記の通りである。

$$L_G := -(C(\tilde{x}) - B) \ln P_{G(z)}(\tilde{x}) \quad (3.11)$$

$$= (C(\tilde{x}) - B) \cdot \text{NLL}_{G(z)}(\tilde{x}). \quad (3.12)$$

ここで、NLL を計算するにあたって、Logits が NLL と同様に対数値になっているため、計算精度を高めることができるため、logits の生成を採用している。実際に Policy を生成するにあたっては、Logits を Sigmoid か Softmax にかける。売りか買いか、新規注文かキャンセルか、マーケットオーダーであるかについては、二値分類のタスクであるので、Logits を生成し、Sigmoid を通じて確率値に落とし込む。それ以外に関しては、他クラス分類になるので、Softmax を通じて確率値に落とし込む。加えて、すべての層で Spectral Normalization [105] を採用している。

一方で、図 3.4 は Critic の詳細について示している。基本的な構造は Generator と同じであるが、最終層付近の層が若干異なる。図 3.4 では Hinge Loss を最終層の後ろにしている例で例示しているが、Hinge Loss を採用しない PGSGAN の場合は、線形層の出力をそのまま Critic の出力として採用する。

22第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配 GAN

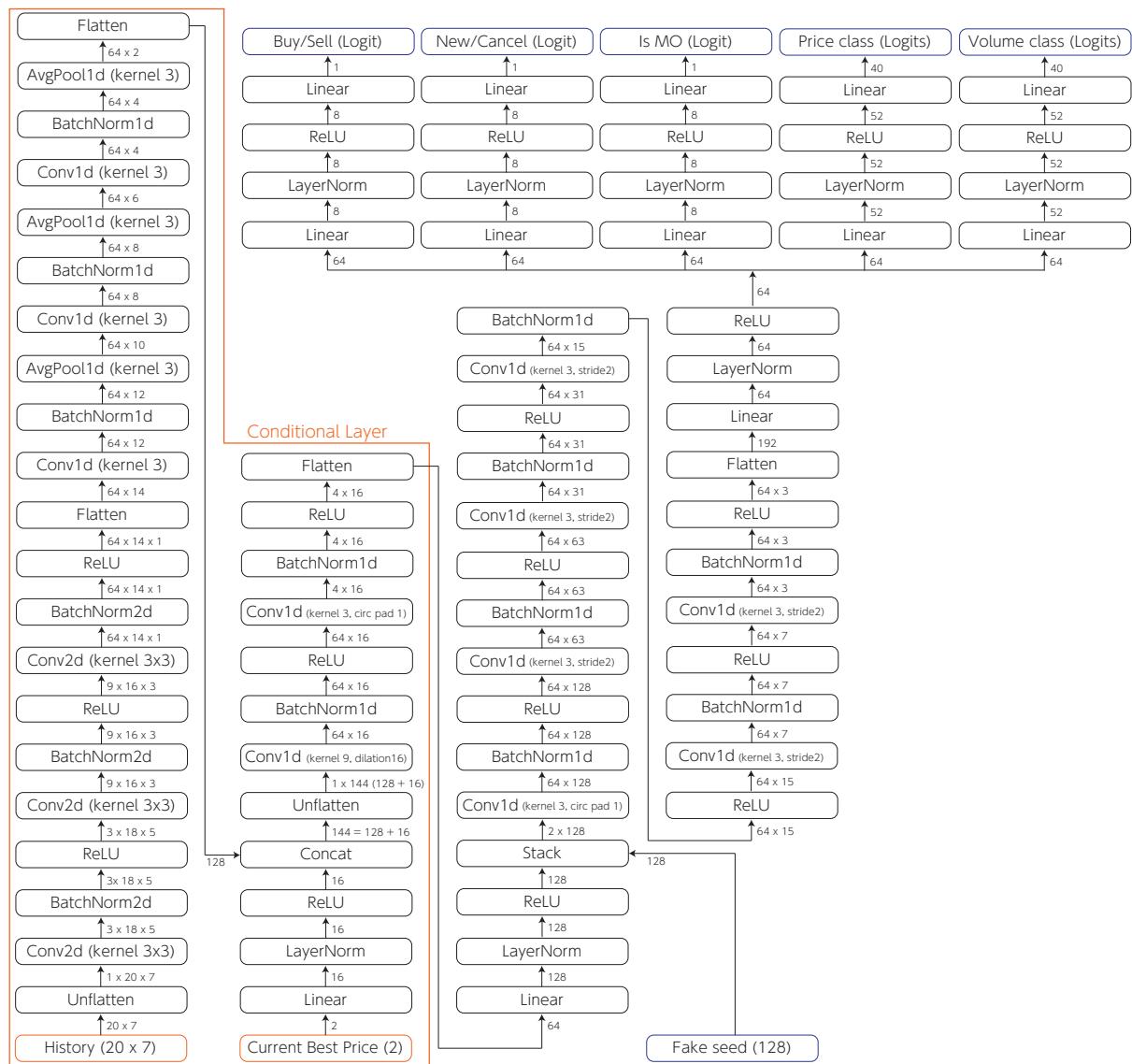


図 3.3: 詳細な Generator の機構

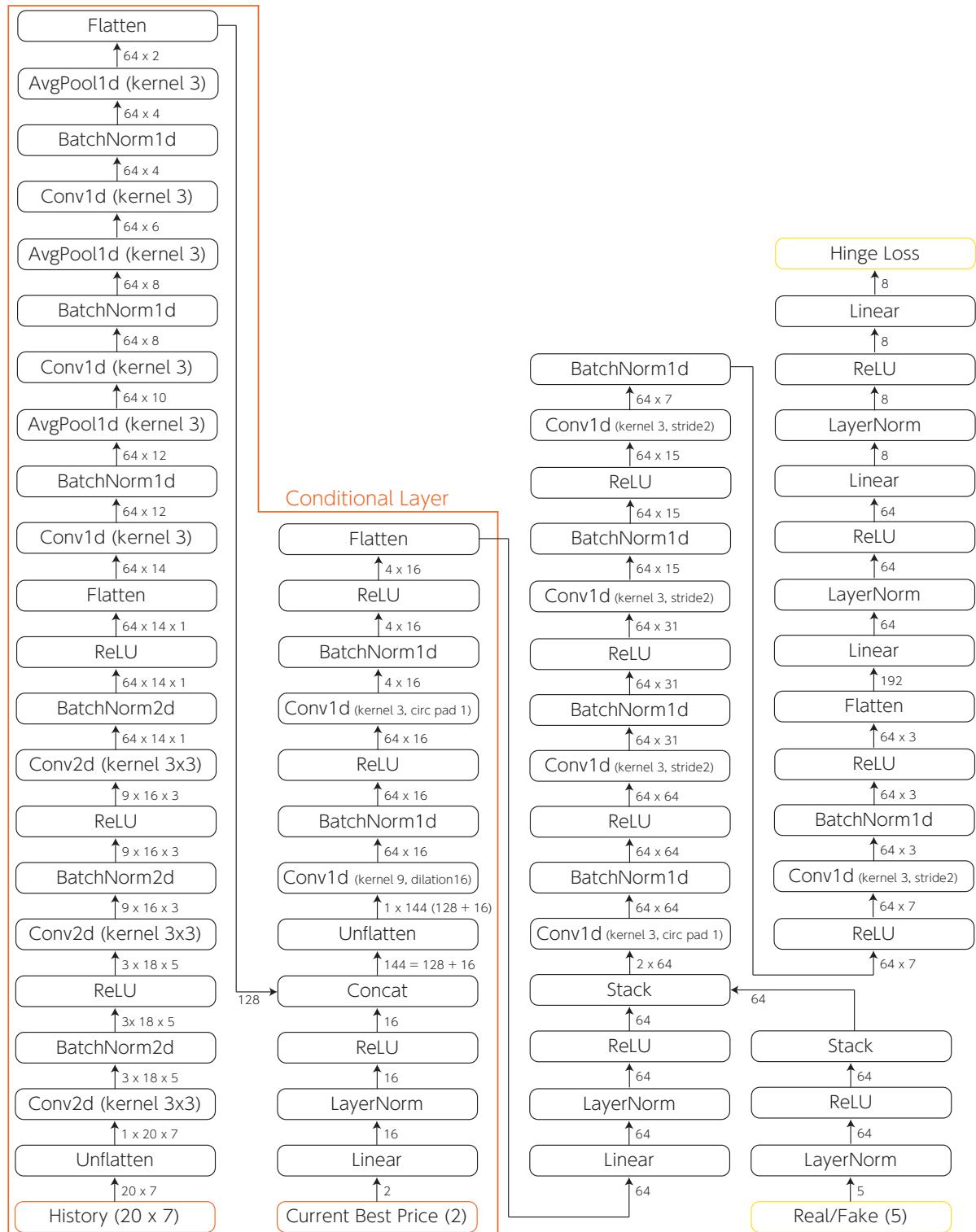


図 3.4: 詳細な Critic の機構

3.3 提案手法2: Policy Gradient Stock GAN with Hinge Loss (PGSGAN-HL)

PGSGAN の亜種として、図 3.4 にあるように、最終層に Hinge Loss を採用したバージョンも作成した。Hinge Loss は、もともと、Geometric GAN [139] で提案された。

Hinge Loss は以下のように定義される。

$$\begin{cases} \max(x + 1, 0) & (\text{critic training for fake data}) \\ \max(1 - x, 0) & (\text{critic training for real data}) \\ x & (\text{generator training}) \end{cases} . \quad (3.13)$$

この層が Critic の最終層として挿入される。

他については、通常の PGSGAN と同じである。

3.4 比較手法

3.4.1 Stock GAN (S-GAN)

Stock GAN (S-GAN) は Li ら [106] により提案され、WGAN-GP [104] に基づいた機構を採用している。S-GAN は LSTM で条件データ（ヒストリカルデータ）を処理し、CNN でその出力を処理する。本研究では、比較モデルとして S-GAN を再現実装し、実験を行った。しかしながら、実験条件を合わせるために、以下の変更を実施している。

- 連続ダブルオーケーション (CDA) ネットワークの削除：CDA ネットワークは、もともと、裁量気配値の更新に使用されていた。これは、前提として、現在の板情報がわからないという仮定、つまり、すべての注文情報を得られないという仮定を置いており、この前提に基づくと、次の注文が生成されたのちの裁量気配値の計算を行うことができない。そのため、その計算の近似のネットワークの構築が必要であるという目的から S-GAN で導入された機構である。しかしながら、本研究ではすべての注文と板が使用可能であり、CDA ネットワークによる近似を行う必要がないため、本研究ではこの部分を削除した。
- Time signal の削除：従来の S-GAN の場合、Time signal という入力の特徴量が存在する。これは、主に、1日のなかでいつの注文時系列なのかを特定するために入力されているものである。そもそも、S-GAN では、24時間注文可能な市場を想定しており、時間的特徴を効率よく組み込むために導入されている。しかしながら、本研究では、東京証券取引所のデータを使用するため、1市場の時間は2.5時間に限られており、それが日に2回ある形であり、そもそも時間帯というほどには長い期間を対象にしていない。そのため、本研究では Time signal を削除することとした。

- 價格シグナルを絶対値から相対値に変更：S-GAN のオリジナルの実装では、注文の特徴量の一つの価格情報が絶対値で入力されている。これは、元の研究では、たかだか 1 日程度の時間スケールでの生成を行っていることから、価格の絶対水準の変化が小さいために、絶対値での入力でも問題ないと推測される。一方で、本研究では、半年を超える長い時間スパンのデータを取り扱うために、絶対値で特徴量を構成すると、価格の絶対水準の変化のために、特徴量の空間がスペースになり、学習が困難になることが想定される。そのため、本研究では、PGSGAN の特徴量構成に合わせて変更を行うこととした。
- 注文間隔の生成の削除：ティックレベルの時間スケールにおいては、注文の到着順番やその間隔は、ランダムである可能性が高い。そもそも同時に複数の行動主体が注文を出している可能性がある。そのため、注文の到達間隔に関しては、それらを考慮して、別途モデリングをすべきであると考える。そのため、PGSGAN と同様に生成対象としないこととした。

他の機構に関しては、元と同じ実装とした。

加えて、S-GAN に関しては、全学習 Epoch 数を半分に設定することにした。詳細な設定は後述する。なお、これは、S-GAN が 1-Lipschitz 制約を達成するために、Gradient Penalty [104] を導入しており、Gradient Pelanty を計算するためには、二度、勾配逆伝播を行わなければならないため、実質的に 1 epoch で 2 epochs 分の計算リソースを必要とするためである。

さらに、S-GAN は、前述の通り、偽注文として、連続値での出力を行う。そのため、評価に当たっては、これを丸め込むことで、PGSGAN と同等の評価を行うこととした。

3.4.2 DCGAN

比較モデルとして、さらに、DCGAN も採用した。S-GAN と同様に連続値で生成を行うモデルであり、基本的なアーキテクチャは CNN ベースとした。また、S-GAN と同様に、偽注文の生成の評価に当たっては生成された注文を同様に丸め込んで評価を行う。

3.5 実験

実験においては、下記で説明する条件を満たす銘柄の中から 10 個をランダムに選択することとした。

まず、本研究では、東京証券取引所のデータを用いているため、東京証券取引所に上場していることが条件である。

そして、次の条件として、日経 225 に採用されている銘柄であることを条件とした。日経 225³は、東京証券取引所における指標の中で、代表的な指標のうちの一つである。日

³<https://indexes.nikkei.co.jp/en/nkave/index/profile?idx=nk225>

経225は、流動性や業種バランスなどを考慮して選択されている銘柄であり、これに含まれていることは、一定の流動性が認められているといえる。本研究においては、Tick時間スケールでの現実的な注文時系列の生成を目的としているため、それなりに流動性が存在し、注文を生成するに足るだけの注文が存在していることが望ましい。そのため、流動性の観点から、日経225に含まれていることを一つの条件として採用することにした。なお、日経225に採用されている銘柄は、毎年10月に定期的に見直しが行われるので、この見直しも後述の条件で考慮に入れることにした。

3つ目の条件としては、TOPIX100に採用されていない銘柄であることを条件とした。TOPIX100も日経225と同様に、東京証券取引所における代表的な指標の一つであり、こちらは、東京証券取引所が自ら指定を行っている⁴。TOPIXインデックスシリーズは、いくつかの指標が存在し、それぞれが異なる。TOPIX100の場合は、時価総額、流動性の観点でどちらもTOP100に相当する程度に大きいことが採用条件である。このTOPIX100に採用されている銘柄に関しては、東京証券取引所では少し特殊な扱いを受ける。このTOPIX100に採用されている銘柄に関しては、取引における呼値がより小さい値に設定されることになっている。そのため、そもそも流動性が大きいことに増して、呼値が小さいことにより、非常に多くの注文が発注される。流動性という観点では本研究に有用ではあるものの、呼値の処理(40クラスでは到底足りなくなってしまう)やデータ量の膨大さゆえに、計算リソースの観点でとても困難である。そのため、本研究では、TOPIX100は対象銘柄から除外することとした。

4つ目の条件が、一定の期間中に常に日経225に含まれており、TOPIX100に含まれていないことを条件とした。この一定の期間は、データ期間よりも長めにとることで、採用・非採用の変化による注文への影響をないようにするために、2018年から2020年の期間とした。この期間に、指定から外されたり、新規に指定されたりすることがないことが条件になる。一般に、新しく指定されることがわかると、取引量が増加することが知られているため、その影響が及ぶ期間をデータに含めないためである。

そして、最後の条件が、データ期間中に呼値の変化がないことである。東京証券取引所においては、呼値は株式価格のレンジごとに設定されている。呼値は $N \times 10^M$ ($N = 1, 3, 5, M = 3, 4, 5, 6, 7$) のいずれかである。この呼値の変化がデータ期間中に起こった場合の処理に関しては、かなり難しい。そのため、本研究では、ひとまず、呼値が一定とし、そのような銘柄だけを対象にすることとした。

データ期間は、2019年の1月から9月の10か月間である。日経225は10月の第一営業日に、TOPIX100は10月の最終営業日に入れ替えが発生するため、この期間を避けた設定とした。

上記の条件に基づいて、81銘柄が抽出された。そもそも日経225に含まれており、TOPIX100に含まれていないという条件の時点で125銘柄に絞られるため、半数以上の銘柄が候補として残ったことになる。

この81銘柄の中からランダムに選んだ10銘柄のデータを実験に採用した。実験に採用

⁴<https://www.jpx.co.jp/english/markets/indices/topix/>

された銘柄は表 3.1 の通りである。

表 3.1: 選ばれた 10 銘柄の概要

銘柄番号	名前	分類 (Bloomberg)	注文データの数
5901 JP	東洋製罐グループホールディングス	Containers & Packaging	6,569,563
5333 JP	日本碍子	Auto Parts	6,077,554
8355 JP	静岡銀行	Banks	5,307,488
5631 JP	日本製鋼所.	Other Machinery & Equipment	6,787,814
9532 JP	大阪瓦斯	Gas Utilities	7,914,464
7012 JP	川崎重工業	Diversified Industrials	9,122,778
2501 JP	サッポロホールディングス	Alcoholic Beverages	4,852,475
4005 JP	住友化学	Basic & Diversified Chemicals	6,319,126
7752 JP	リコー	Consumer Electronics	6,942,513
7911 JP	凸版印刷	Printing Services	6,057,922

これらは、時系列の順に 8:1:1 に分割し、Train, Valid, Test データとして使用する。

実験のタスクとしては、次の注文生成を行う。長期時系列の生成も重要なタスクであるが、まずは、次の注文生成がもっとも基本的な単位であることから、次の注文生成だけにフォーカスして実験・評価を行う。次の注文生成が精度良く行うことができれば、その繰り返しで長期時系列を生成できることから、次の注文生成に大きくフォーカスすることは一定の妥当性があると考える。

PGSGAN の挙動を確認するために、実験では、生成された Policy における実注文の負の対数尤度 (NLL) と Policy のエントロピー (情報量エントロピー) を計算することにした。NLL は $NLL_{G(z)}(x)$ と定義することができる。ここで、 x は実際の次の注文である。これは、Generator が生成した Policy がどの程度実データにフィットしているのかということについて評価することができる。NLL が低ければ、それは実データに対するフィットが良いことを示す。しかしながら、完全に実データにフィットしている状態は過学習とも言える。GAN は完全に実データのみを生成するような状態になってしまふと、それはもはや Mode Collapse を起こしていると言えるので、適切な状態とはいえない。そのため、NLL が低いことは常に良いことではないく、ある程度低くなつていれば充分であると言えるであろう。これらを踏まえて、実験では、あくまで NLL を学習が適切に進んでいることの確認のツールとして考え、最小の NLL は追求しないこととする。理論的には、ランダムで生成を行なった場合の By-chance NLL は

$$-\ln \{1/(2 \times 2 \times 2 \times 40 \times 40)\} \approx 9.45 \quad (3.14)$$

より、9.45 である。一方で、エントロピーは以下の通りに定義される。

$$H(G(z)) := \sum_{x \in \mathbb{X}} -p_{G(z)}(x) \log_2 p_{G(z)}(x). \quad (3.15)$$

ここで、 \mathbb{X} は全ての生成しうる注文パターンである。エントロピーは、生成された Policy の偏り具合を見ることにより、どの程度学習が進んだかということの確認に使える。もし、充分に学習が進んでいたとすれば、生成された Policy に偏りが発生していく。エントロピーの By-chance の値を計算すると、13.64 であるとわかる。

$$\sum_{x \in \mathbb{X}} -\frac{1}{2 \times 2 \times 2 \times 40 \times 40} \log_2 \frac{1}{2 \times 2 \times 2 \times 40 \times 40} \quad (3.16)$$

$$= \log_2 (2 \times 2 \times 2 \times 40 \times 40) \approx 13.64. \quad (3.17)$$

ここで、 \mathbb{X} は全ての生成しうる注文パターンである。この数値からどの程度エントロピーが下がったのかということを一つの学習の指標として採用する。

生成の評価としては、生成された注文と実際の注文の分布の比較を行う。これは、PGSGAN 以外のすべての比較モデルでも実施する。基準としては、全生成クラス ($2 \times 2 \times 2 \times 40 \times 40$) に対して、Kullback–Leibler divergence (KLD) [140] と Mean Square Error (MSE) で比較を行う。Kullback–Leibler divergence は以下の通り定義される。

$$D_{KL}(P, Q) = \sum_{x \in \mathbb{X}} P(x) \log_2 (P(x)/Q(x)). \quad (3.18)$$

30第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配 GAN

ここで, \mathbb{X} はすべての注文クラス, $P(x)$ は実データの注文クラス x に対する確率, $Q(x)$ が実データの注文クラス x に対する確率である. $P(x) = (Q(x) =) 0$ の場合は $P(x) \log_2 \frac{P(x)}{Q(x)}$ は 0 として計算する. $\exists x, P(x) \neq 0, Q(x) = 0$ となる場合については, KLD は無限となる. KLD が無限になるのは, Mode Collapse が Generator で発生した場合などの特殊な場合に限られる. さらに, DCGAN と S-GAN については, 生成された連続空間に配置された注文を PGSGAN と同じ形式に丸め込むことによって評価を行う. また, GAN は入力に含まれる乱数の影響を受けるため, この影響を低減する目的で, テストにおける同一局面で 100 個の異なる乱数を用いて生成を行い, その平均をとることとした.

さらに, GAN の実験に当たって採用した他のパラメータとしては, バッチサイズが 2048, 学習 epochs が 5000 epochs, 学習率 (Generator, Critic 双方) が 10^{-5} であり, オプティマイザーには Adam を採用した. 加えて, 学習の安定性のために, Generator と Critic の学習の比率 (TTUR; Two Time-scale Update Rule [141]) を 1 : 5 に設定した. また, 評価に多くの計算量を消費するのは効率的ではないため, 評価は 10 epochs 間隔で実施することとした.

表 3.2: すべての結果一覧 (KLD)

Ticker	PGSGAN (KLD)	PGSGAN-HL (KLD)	S-GAN (KLD)	DCGAN (KLD)
5901 JP	0.208467 ± 0.000107	0.168823 ± 0.000128	∞ ± NaN	∞ ± NaN
5333 JP	0.257479 ± 0.000121	0.203760 ± 0.000121	∞ ± NaN	∞ ± NaN
8355 JP	0.136612 ± 0.000071	0.139955 ± 0.000061	∞ ± NaN	∞ ± NaN
5631 JP	0.215376 ± 0.000064	0.209126 ± 0.000077	∞ ± NaN	∞ ± NaN
9532 JP	0.198947 ± 0.000053	0.190806 ± 0.000078	∞ ± NaN	∞ ± NaN
7012 JP	0.150013 ± 0.000067	0.138801 ± 0.000096	∞ ± NaN	∞ ± NaN
2501 JP	0.243355 ± 0.000108	0.216768 ± 0.000173	∞ ± NaN	∞ ± NaN
4005 JP	0.136136 ± 0.000053	0.144642 ± 0.000071	∞ ± NaN	∞ ± NaN
7752 JP	0.164498 ± 0.000036	0.123295 ± 0.000046	∞ ± NaN	∞ ± NaN
7911 JP	0.228341 ± 0.000083	0.203080 ± 0.000107	∞ ± NaN	∞ ± NaN

表 3.3: すべての結果一覧 (MSE)

Ticker	PGSGAN (MSE)	PGSGAN-HI (MSE)	S-GAN (MSE)	DCGAN (MSE)
5901 JP	0.000479 ± 0.000003	0.000386 ± 0.000003	0.019759 ± 0.000053	0.148968 ± 0.00005
5333 JP	0.002164 ± 0.000002	0.000945 ± 0.000005	0.004611 ± 0.000011	0.143241 ± 0.00007
8355 JP	0.000505 ± 0.000002	0.000392 ± 0.000001	0.012347 ± 0.000018	0.093950 ± 0.000016
5631 JP	0.000499 ± 0.000002	0.000432 ± 0.000003	0.027200 ± 0.000035	0.107452 ± 0.000026
9532 JP	0.000300 ± 0.00000	0.000278 ± 0.000003	0.030383 ± 0.000029	0.145481 ± 0.000011
7012 JP	0.000332 ± 0.000003	0.000257 ± 0.000003	0.020080 ± 0.000042	0.145100 ± 0.000003
2501 JP	0.000428 ± 0.000002	0.000391 ± 0.000003	0.014947 ± 0.000039	0.146345 ± 0.000012
4005 JP	0.000512 ± 0.000004	0.000393 ± 0.000003	0.014715 ± 0.000017	0.109629 ± 0.000009
7752 JP	0.000866 ± 0.000003	0.000691 ± 0.000004	0.009458 ± 0.000015	0.141400 ± 0.000005
7911 JP	0.000580 ± 0.000003	0.000473 ± 0.000004	0.019275 ± 0.000029	0.107608 ± 0.000012

3.6 結果

表3.2, 3.3は、それぞれ、KLDとMSEでの評価の結果である。それぞれの行は前述のランダムに選ばれた10個の銘柄に対応している。最も良い結果は太字で表記している。S-GANとDCGANは上述のとおり、KLDにおいて無限になってしまっている場合が存在する。

まず、KLDに関しては、銘柄によるが、PGSGANとPGSGAN-HLが高い性能を発揮している。これは、提案手法がS-GANとDCGANを上回る性能を達成しているということである。一方で、MSEに関しては、すべてのモデルが有限の評価値を得ている。この結果によると、PGSGAN-HLがどの銘柄に対しても最高の性能を示している。加えて、PGSGANは、PGSGANよりも性能が悪いものの、S-GANとDCGANを上回る性能を示した。

また、先行研究で示されていた通り、S-GANはDCGANを上回る性能を示しており、先行研究との合致性も確認できた。ただし、提案手法と比べると、S-GANの性能は極めて限られているともいえる。

さらなる評価のために、生成された注文の分布に関する詳細な調査を行った。図3.5–3.14では、各銘柄における生成された注文の分布を比較している。

これらの結果に基づくと、特にDCGANにおいて、数量と価格の分布の再現に失敗していることがわかる。

加えて、S-GANでは、価格や数量における、小さな山をとらえることに失敗していることや、丸め込みを行うことによる、ピークの若干のずれなどが顕著に観測できる。特に、S-GANの場合、PGSGANやPGSGAN-HLよりも、分布が滑らかになっており、現実の特徴と乖離が確認できる。

3.7 考察

まず、提案手法は先行研究を超える性能を発揮している。PGSGAN-HLとPGSGANが先行研究のモデルの性能を超えたということは、Policy Gradientを用いた提案手法が正しく機能していることを示していると言える。

金融市場における実際の取引ルールを考慮に入れれば、注文の生成において、離散性を考慮に入れることは連続空間で生成するよりも適切であると言えると考えられる。無論、価格や数量は連続値として考えることもできなくはないものの、実際には離散的であるということを考慮することが、注文の生成の性能にも影響した可能性があると考えられる。

この離散性の考慮に当たっては、GeneratorとCriticの勾配の切断問題をPolicy Gradientを用いて解決した。前述の通りPolicy Gradientを用いることで、理論的には実現可能であると示したが、実際に実験を通じて、このPolicy Gradientで問題が解決できることが確認できた。

34第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配 GAN

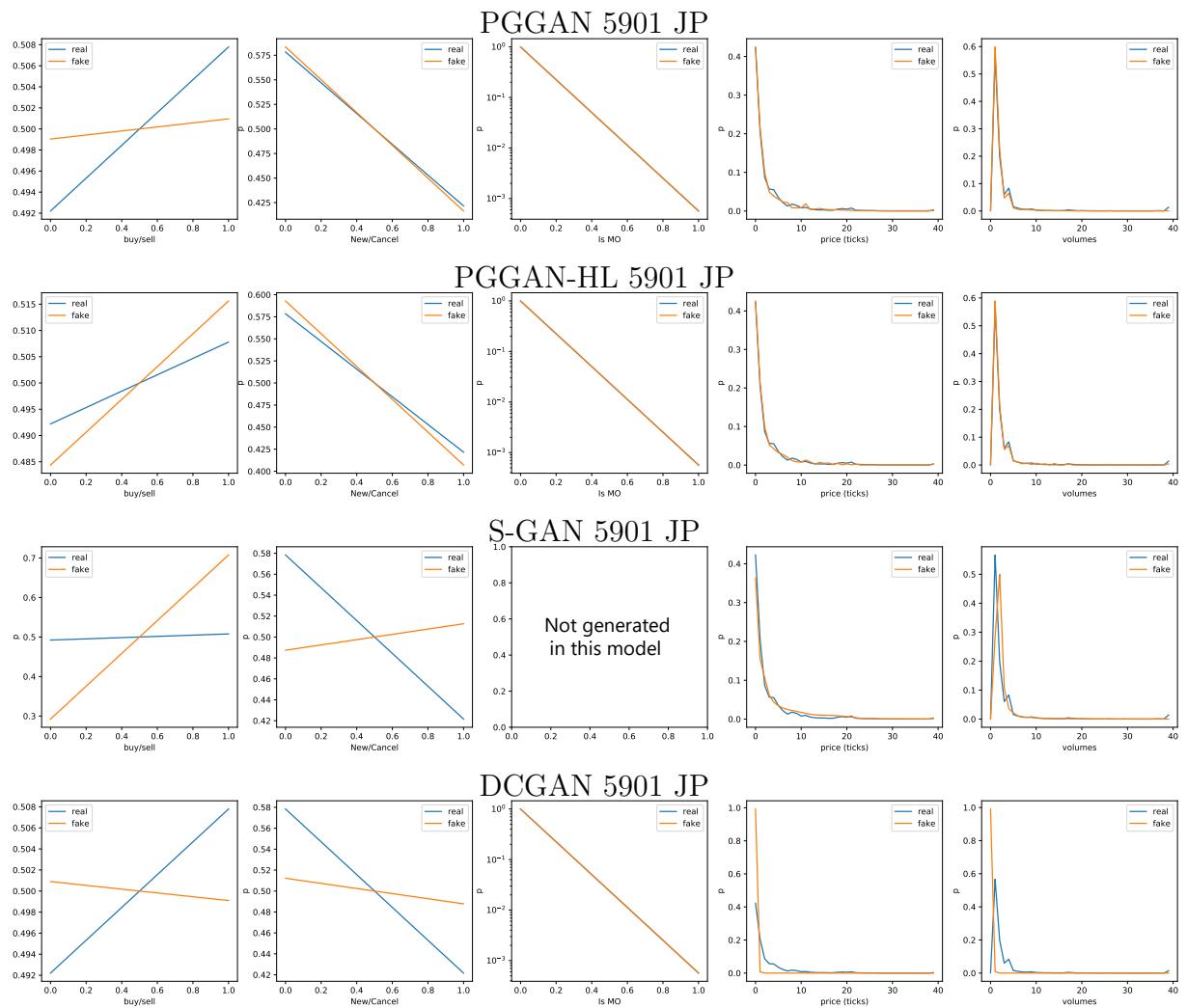


図 3.5: 実データと生成された注文の分布の比較 (5901 JP). 青い線が実データ、オレンジが生成データである。各行はそれぞれのモデルに対応する。

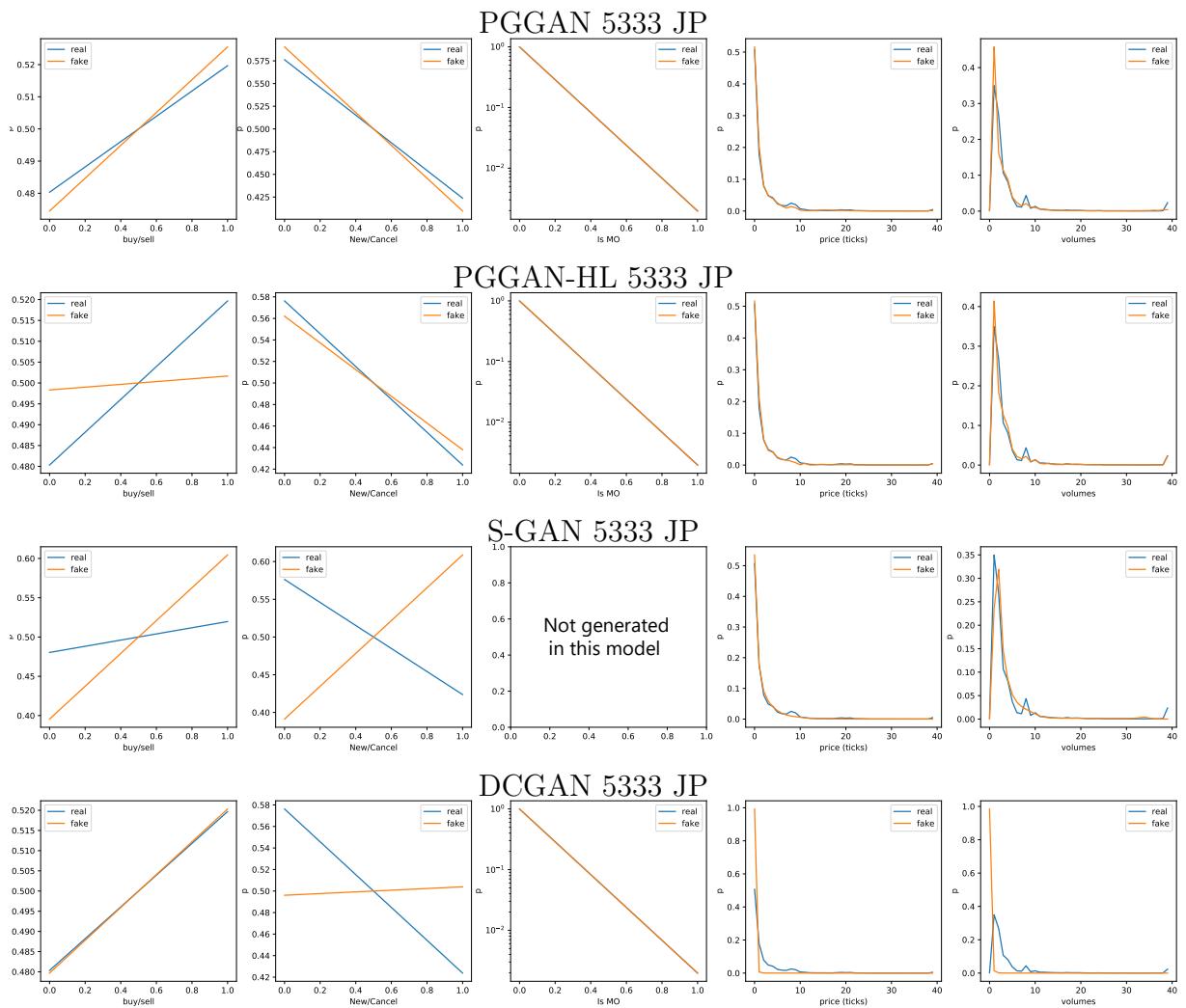


図 3.6: 実データと生成された注文の分布の比較 (5333 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

36第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配 GAN

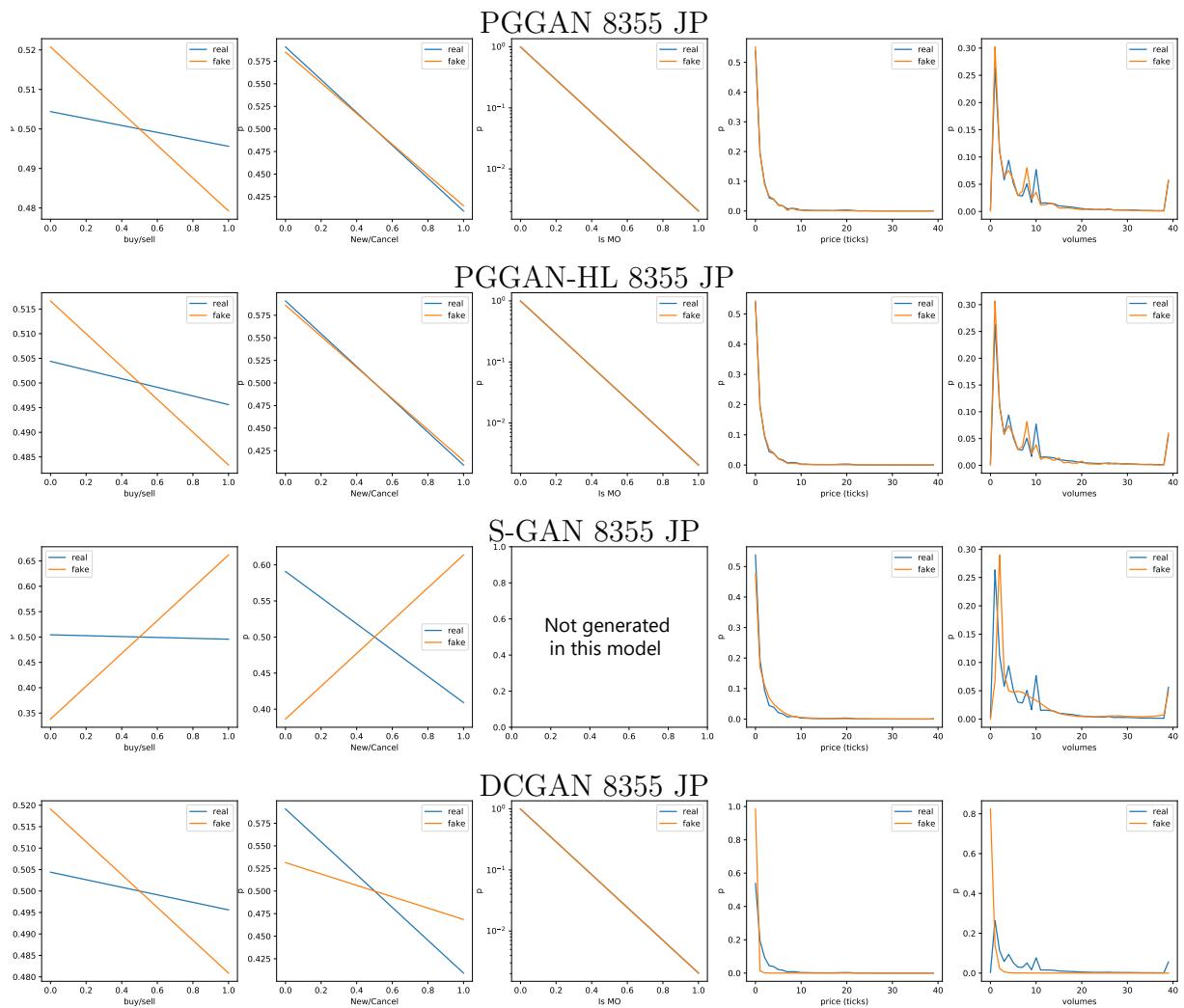


図 3.7: 実データと生成された注文の分布の比較 (8355 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

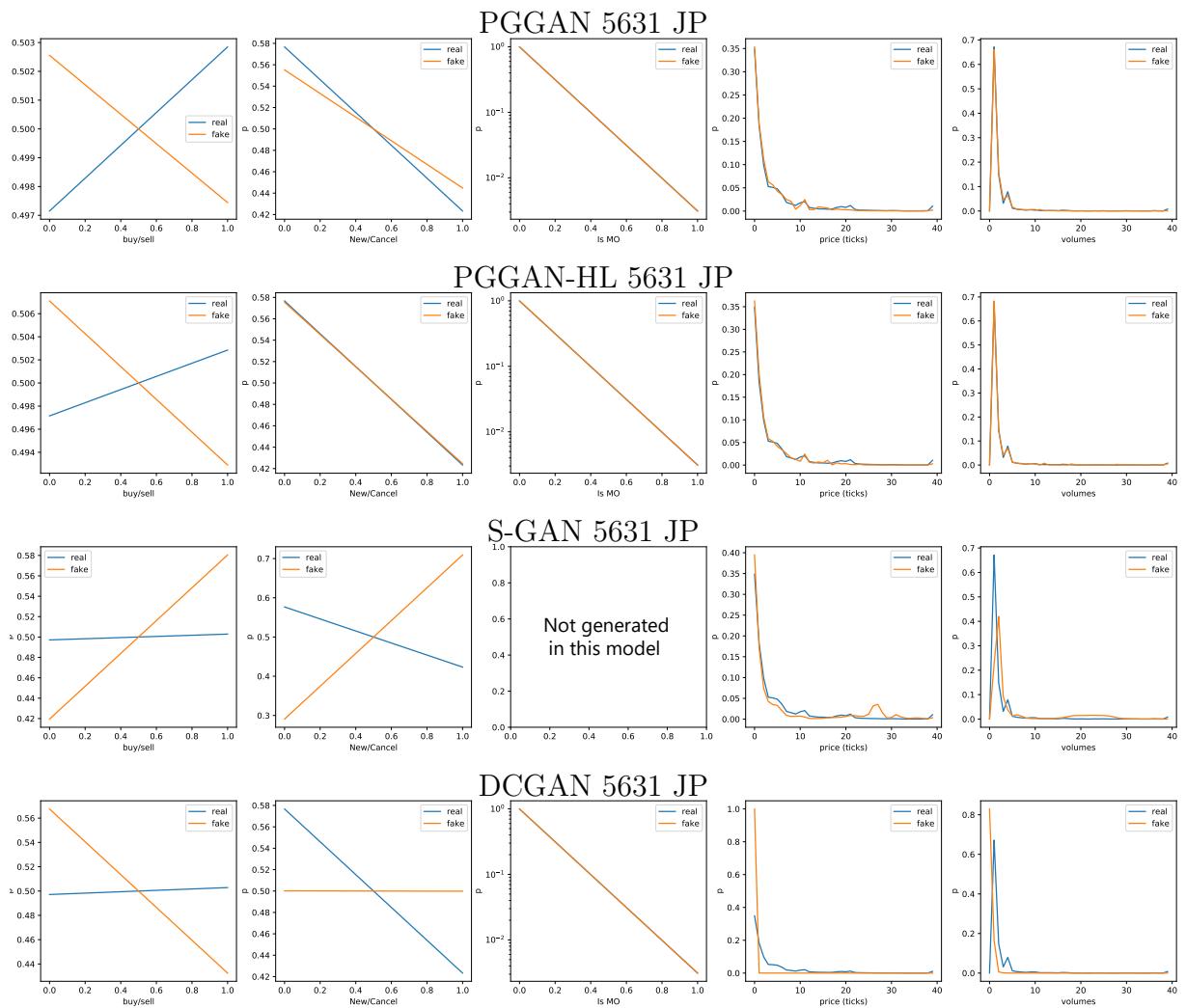


図 3.8: 実データと生成された注文の分布の比較 (5631 JP). 青い線が実データ、オレンジが生成データである。各行はそれぞれのモデルに対応する。

38第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配 GAN

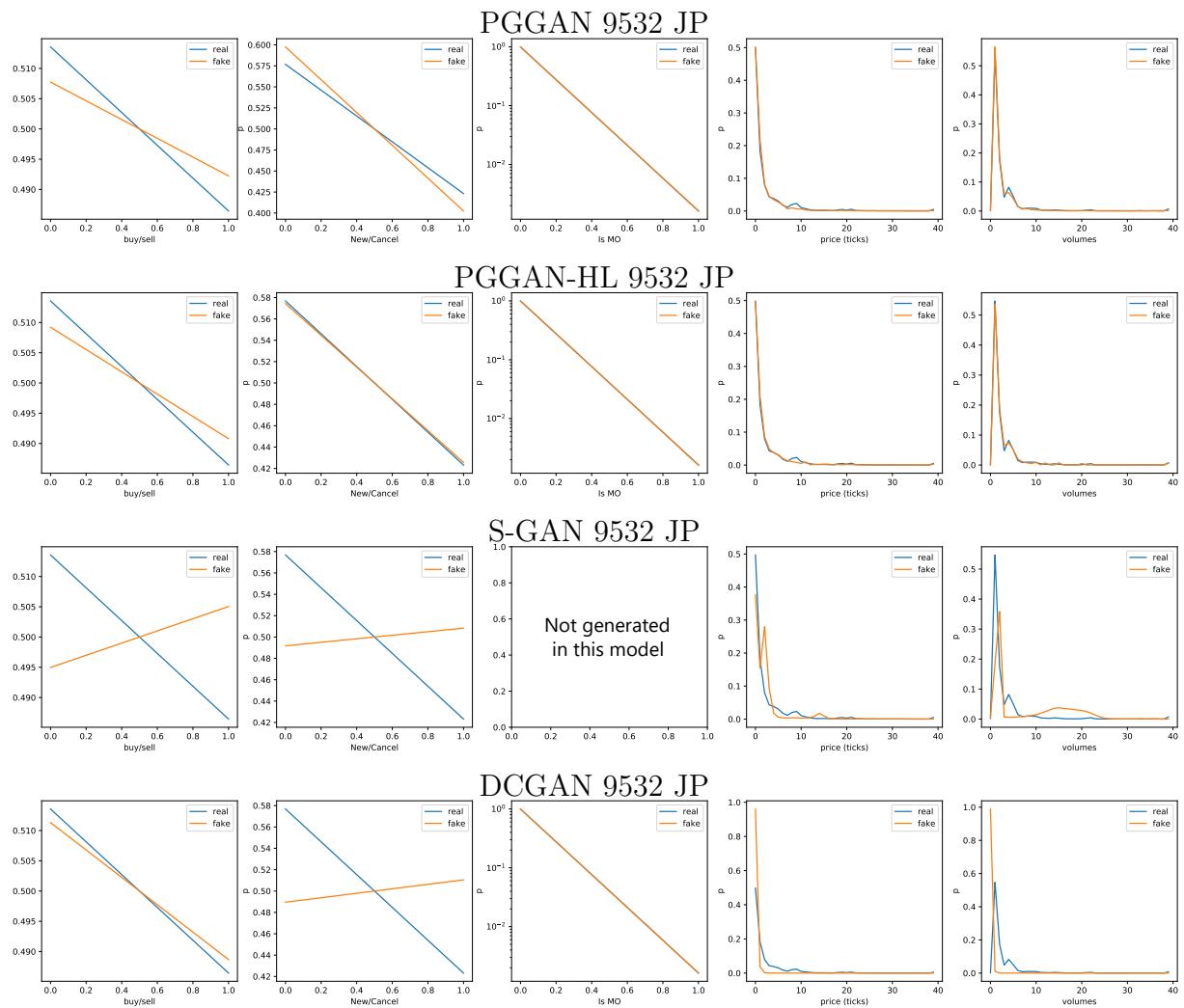


図 3.9: 実データと生成された注文の分布の比較 (9532 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

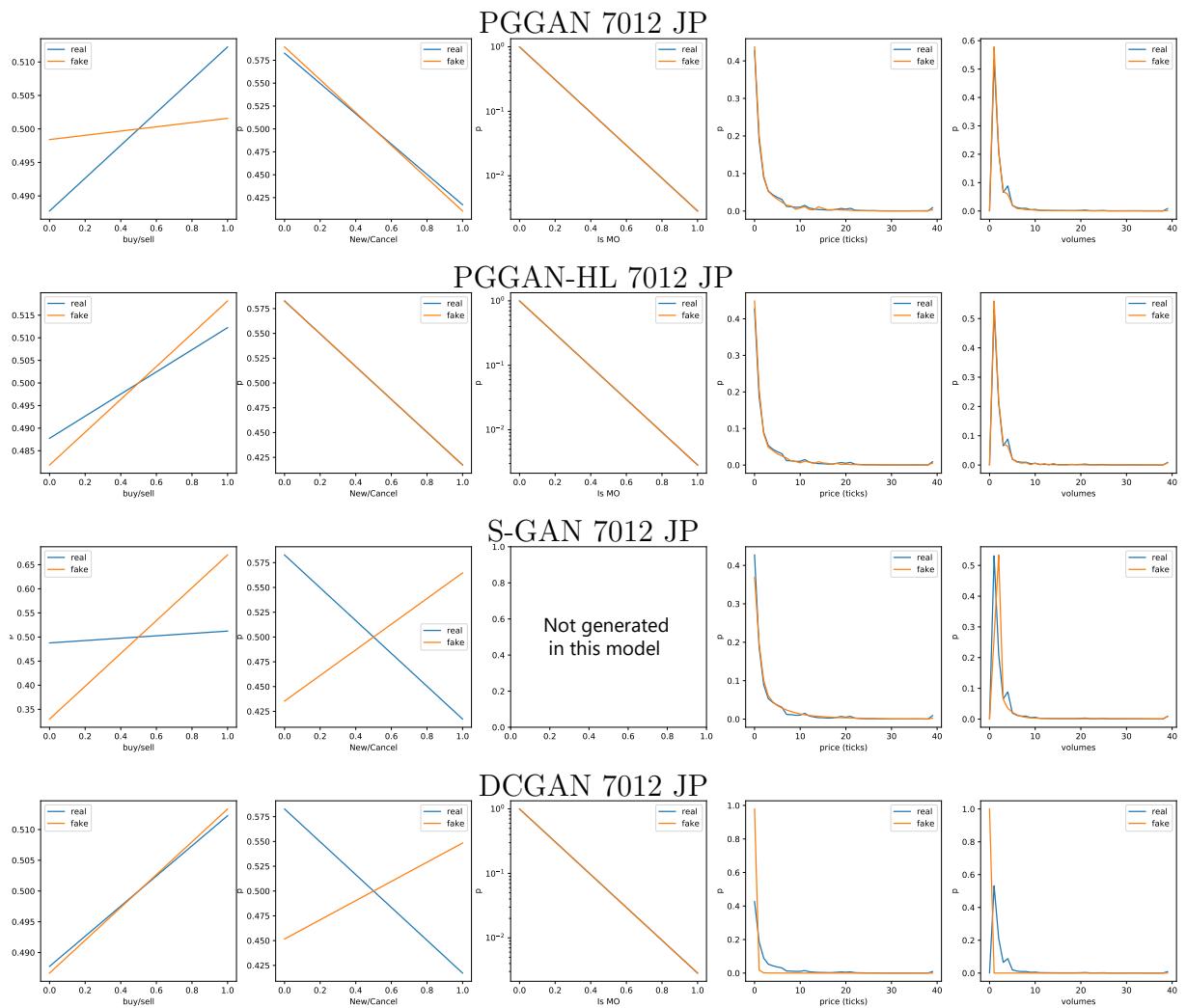


図 3.10: 実データと生成された注文の分布の比較(7012 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

40第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配 GAN

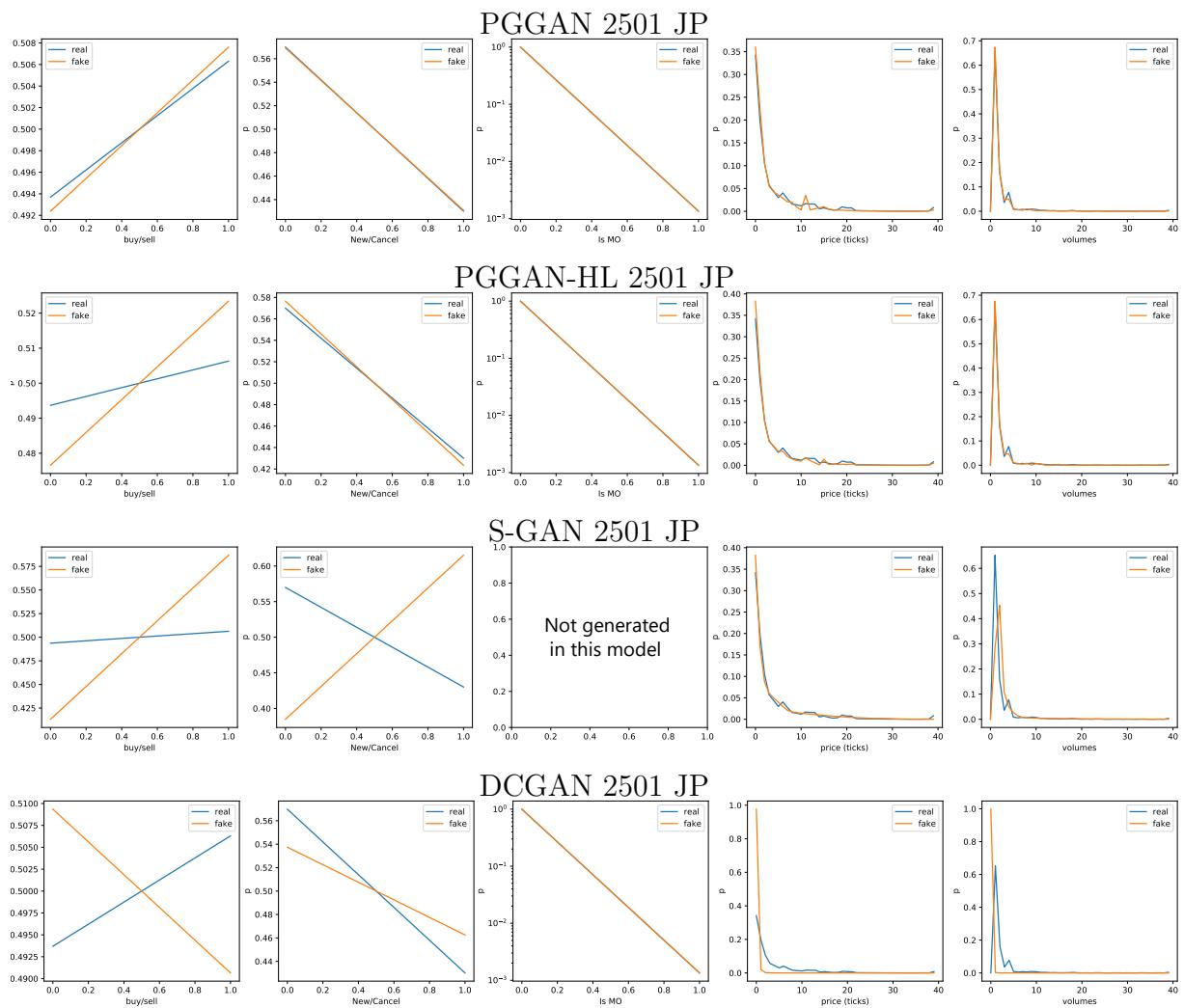


図 3.11: 実データと生成された注文の分布の比較(2501 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

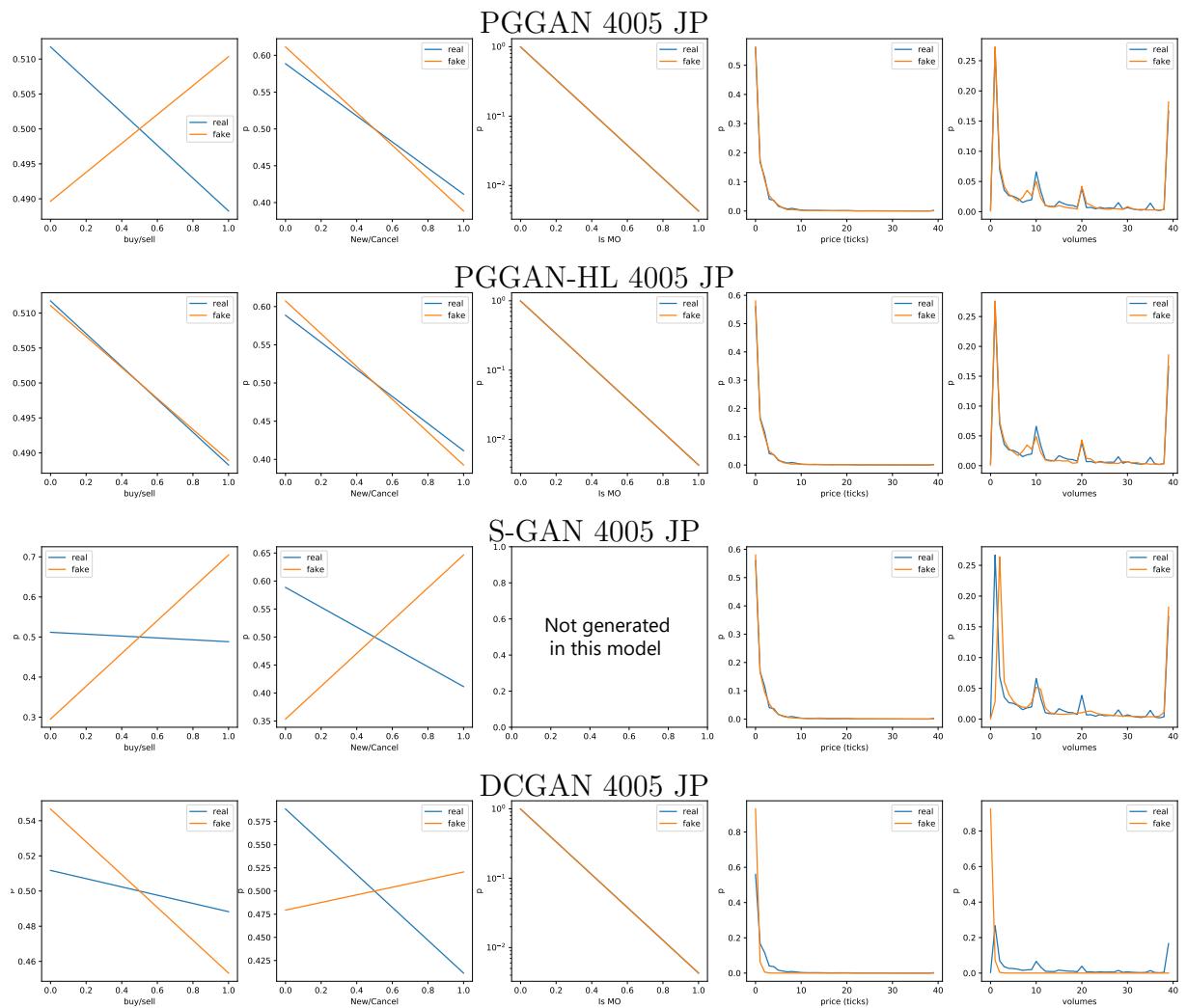


図 3.12: 実データと生成された注文の分布の比較 (4005 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

42第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配 GAN

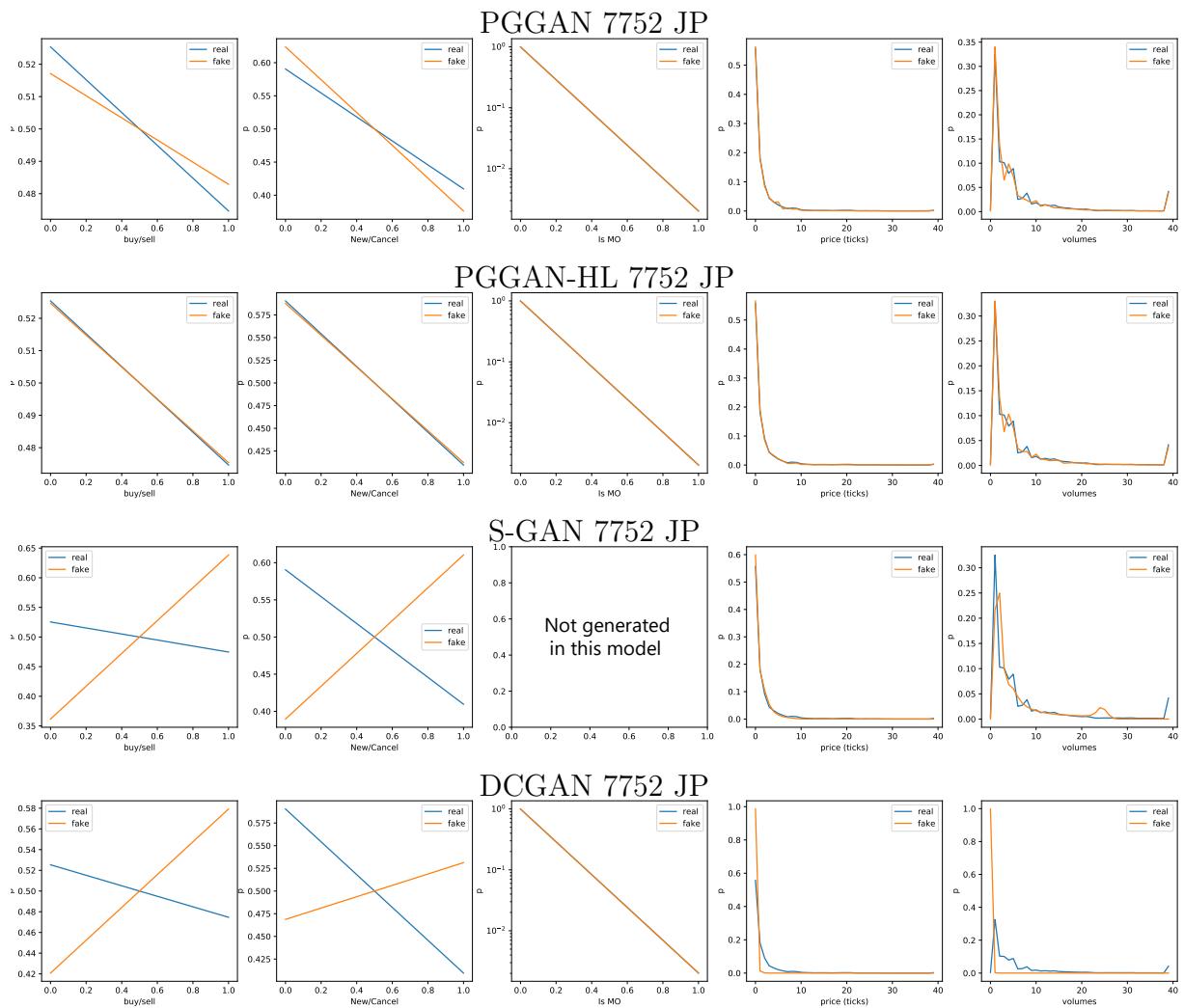


図 3.13: 実データと生成された注文の分布の比較(7752 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

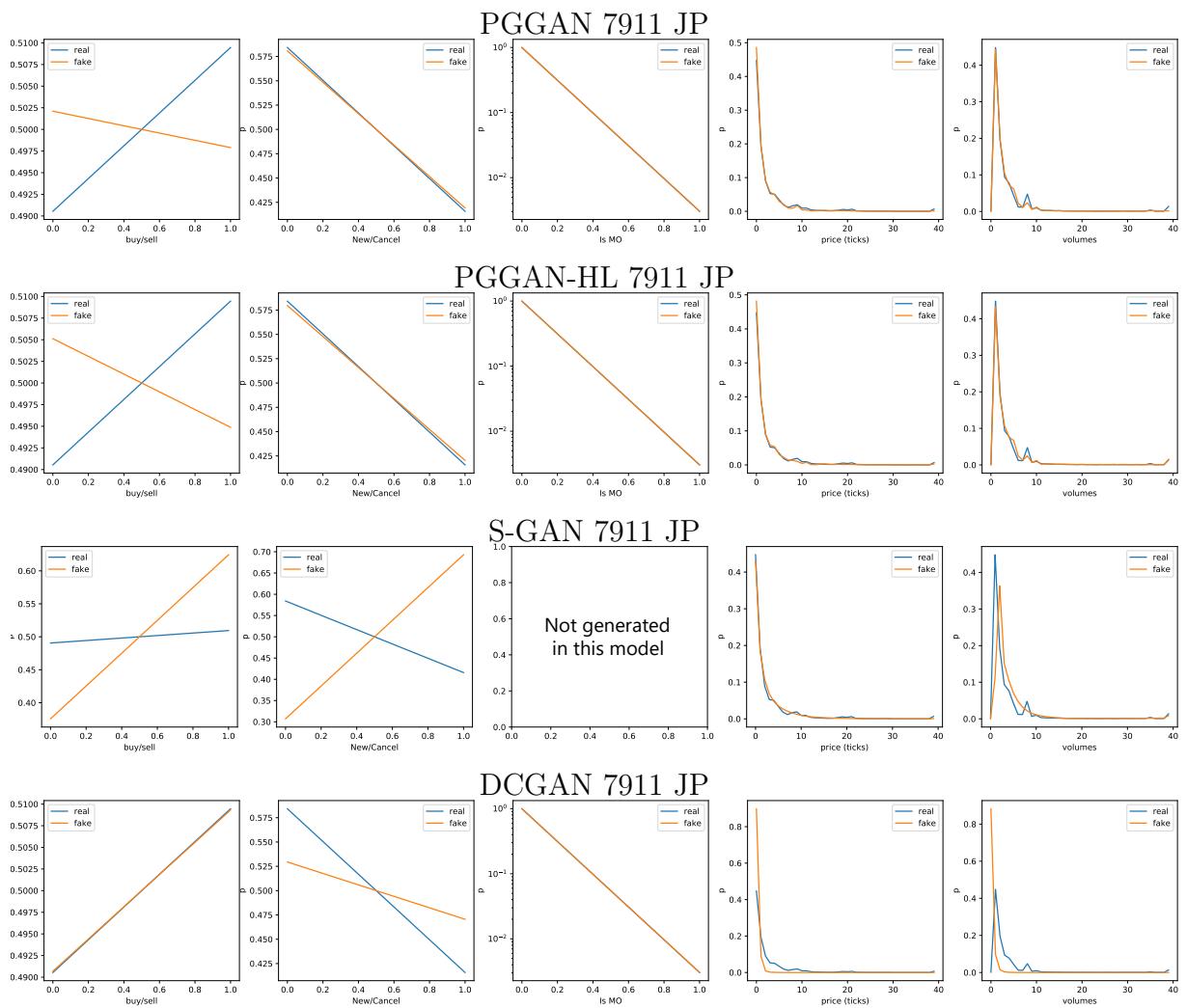


図 3.14: 実データと生成された注文の分布の比較(7911 JP). 青い線が実データ, オレンジが生成データである. 各行はそれぞれのモデルに対応する.

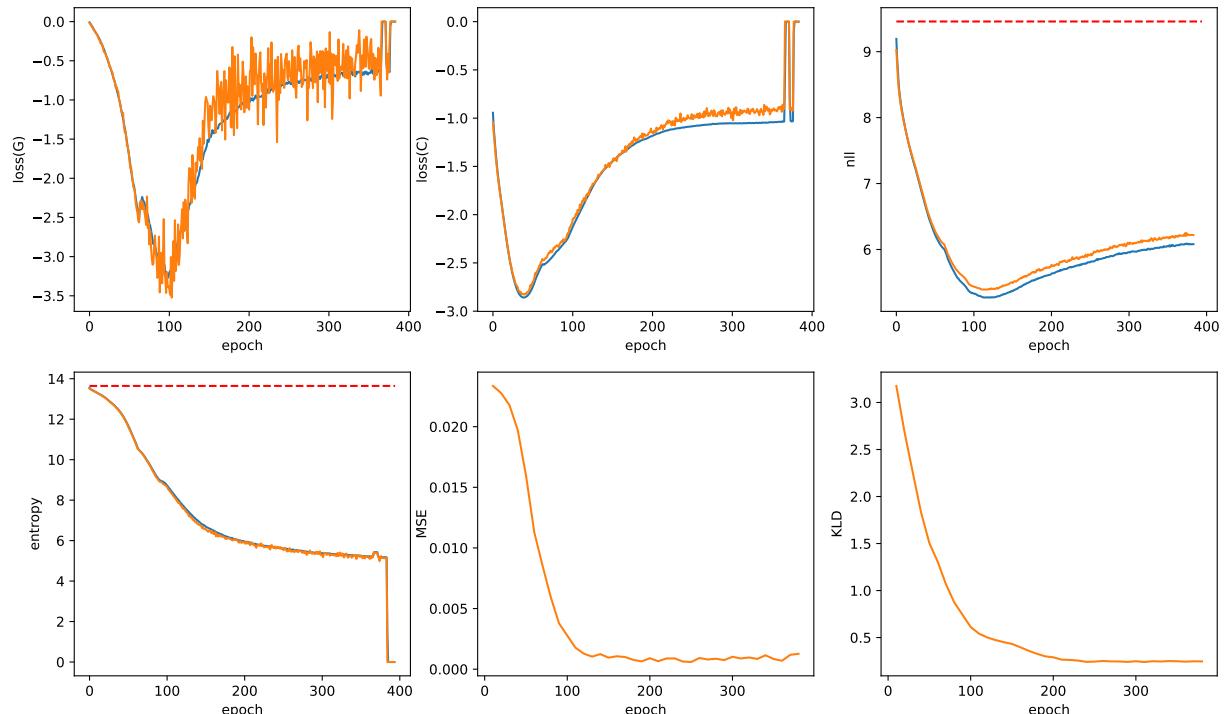


図 3.15: 学習における各種指標の遷移 (PGSGAN 5901 JP). 左上から右に, Generator の損失関数, Critic の損失関数, NLL ($NLL_{G(z)}(x)$). x は実注文である. 下段は, 左から右に, Generator の生成する Policy のエントロピー, 実データと偽データの分布の MSE, 実データと偽データの KLD である. 横軸は epochs である. 青い線が学習における指標, オレンジが Valid データにおける指標の結果である. MSE と KLD に関しては, Valid データのみで計算されている. さらに, 赤い破線は, それぞれの指標における By-Chance レベルの数値を示している.

前述の通り、生成された Policy のエントロピーを学習状況の指標として用いることができると言えられる。この点を実験で確認するために、各種指標をプロットしたものを見たところ有用であるものの、Generator と Critic は敵対的な学習を行っているため、それらの損失は相対的であり、必ずしも正しく状況をとらえられるわけではない。他の一般的な GAN では、学習状況をモニタリングするにあたって、異なる外部タスクを活用する例が存在する。例えば、画像生成においては、Inception Score [142] や Fréchet Inception Distance [141] というような外部のタスクを活用して現在の学習状況をモニタリングする手法が提案されている。しかしながら、評価のために外部学習タスクを活用するのは一般に難しく、金融市場においてはそのタスク自体の準備も容易ではない。図 3.15 によると、エントロピーの遷移は MSE や KLD の遷移と近い挙動を示していることがわかる。そのため、生成された Policy のエントロピーは学習の進み具合の指標としては活用可能である可能性が示唆される。

一方で、生成された注文の分布が実データに近いことは、良い生成の必要条件であるが、十分条件ではない。すでに結果で示した通り、PGSGAN はより現実に近い偽注文の分布を生成することに成功している。しかしながら、Mode Collapse が発生しているなど、単に現実の注文を生成し続けたり、局面を考慮に入れずに実データ全体の分布に基づいて偽注文を生成しているなどの状況は生成器としては良くない。また、生成も入力乱数に寄るため、本研究では異なる 100 個の乱数を入力することで、テストにおいての評価値を安定させるようにしている。しかし、もし、生成器が乱数に基づいて、様々な妥当な注文を生成する能力を有しているとすれば、異なる乱数に対して異なる生成を行うことを検証することを確認すべきであろう。

乱数によって異なる生成を行うことができているかという観点に関しては、NLL を指標として評価を行うことができる。もし、NLL がほぼ By-chance レベルの値であれば、Generator は意味のないランダムな生成をしているということを意味するので、良い生成とはいえないであろう。一方で、NLL が 0 であるのであれば、単に実際の注文を常に出し続ける、ある種の過学習的な状況になっており、多様な妥当な注文を生成するという目的を達成できていない可能性が示唆される。加えて、いかなる乱数を入力にとっても、NLL がそれぞれの局面で一意に定まるとは限らない。それは、入力乱数に依存していないことになるため、Mode Collapse が発生している可能性が否定できない。そのため、本研究では、100 回の異なる乱数における生成を評価する際に、NLL の分散の平均を計算することでその変化具合をとらえることとした。この評価は NLL の計算が可能である PGSGAN と PGSGAN-HL でのみ可能である。実際に計算を行ったところ、NLL の分散は、おおよそ 2.5 から 3.6 程度（銘柄による）であることが分かった。NLL の平均がおおよそ 5 から 6.5 であることを考慮に入れれば、充分に NLL はばらついていることがわかる。また、NLL の絶対水準としても、それなりに高い水準であり、充分に多様性のあるような生成が達成されており、Mode Collapse のような現象は起きていないであろうことが確認できた。

これらの結果を踏まえると、提案手法は、良い Generator としても十分であると言える

46第3章 PGSGAN: 金融市場における現実的な注文データ生成のための方策勾配GAN

であろう。無論、より良いモデルを構築する余地はあるとはいえるが、先行研究よりは良いモデルであるというに足りる結果となっていると考えている。

次に、PGSGAN-HLが多くの実験でPGSGANよりも良い性能を出したことについて考察する。この要因は、PGSGAN-HLとPGSGANの差分であるHinge Lossの導入にあると言えるであろう。PGSGAN-HLと異なり、PGSGANの場合、学習の途中でGeneratorの勾配が0になる現象が確認できている。式3.10で既に説明した通り、Generatorの勾配は、Criticの出力に線形で比例した形式で表される。そのため、GeneratorがCritic以上により精度良く偽注文を生成することができれば、Criticの出力は0になる。それゆえに、Generatorにおける勾配消失が発生してしまい、学習がそれ以上進めなくなってしまう、途中終了せざるを得なくなってしまう。この現象は図3.15で明確に確認できる。一方で、Hinge Lossを導入することで、この問題を回避することができる。その結果として、PGSGAN-HLの方がより長い期間学習を行うことができるため、より良い性能を達成できたと考えられる。

さらに、学習にかかった時間についても考察を行う。本研究の設定においては、それぞれのモデルの学習にかかった時間は概ね以下である：

- PGSGAN: 3–5日
- PGSGAN-HL: 約20日
- S-GAN: 約6ヶ月（推定値。実験では半分のepoch数で実験を行い、それに3ヶ月を要した。）
- DCGAN: 約10日

これらの時間は、データサイズや計算リソースにも依存するので、大まかな時間である。ただ、正確ではないとはいえるが、大まかなモデルの学習難易度を把握するのには役に立つと思われる。これらの数値は、Geforce RTX 2080, 2070super, 2080TiなどのNVIDIAのGeforce RTX 20シリーズの高性能モデルを用いたときの数字である。

これらに基づくと、S-GANは非常に学習難易度の高いモデルであるとわかる。最大の問題は、S-GANが1-Lipschitz制約を満たすためにGradient Penalty [104]を使用していることがある。前述の通り、損失の計算に当たって、Gradient Peanltyは、二度の勾配逆伝播が必要するために、計算量がおよそ二倍になっている。ただし、計算量が二倍になった分、学習epochsを半分にしてもなお、学習に大幅な時間を要する。この問題は、不適切な畳み込み層によるものであり、畳み込み層のカーネルサイズが非常に大きいために、CNNの計算効率面での効率性をつぶしてしまっていることにあると考えている。

一方で、PGSGANの学習速度が圧倒的に早かった理由は、明示的な学習の終了にあると考えられる。PGSGAN-HLと異なり、Generatorの勾配が学習の途中で0になり、そこで学習が終了することは前述のとおりである。これにより、ある程度Generatorの学習が進んだ時点で学習が終了するため、ほどよい箇所で学習が止まることにより、計算時間の

短縮が図られているという状況である。ただし、計算コストと性能のトレードオフの観点からはかなりバランスがよいともいえるであろう。

今後の課題としては、より高性能なGANの追求や、MSEやKLDよりもより適切な評価手法の追求などがあげられるであろう。加えて、本研究では、次の注文の生成というとともに単純なタスクにだけ着目していたが、より長期の時系列の生成と評価も今後の課題であると考える。

3.8 本章のまとめ

本章では、金融市場におけるより現実的な注文の生成を行う新しいGANを提案した。先行研究においては、GANの学習理論の限界から、注文の生成にあたって、注文の特徴量を連続的なものとして取り扱っていた。しかしながら、実際の取引ルールを考慮に入れると、注文の各種特徴は離散的である。例えば、価格や数量でさえも最小単位が存在する。加えて、注文タイプや売り買いなども連続値で表すのは不適切である。そのため、本章では、偽注文の生成に当たって、この離散性を考慮に入れることとした。しかしながら、通常のGANの学習理論では離散性を考慮に入れるてしまうと、Generatorの勾配が取れなくなってしまうため、実現が困難である。そこで、本章では、深層強化学習で比較的よく使われるPolicy Gradientを導入し、この学習を実現した。提案手法では、Generatorが注文分布のPolicyを生成し、Policyに基づいて次の偽注文を生成し、その注文をCriticが実注文と判別するという機構を提案した。実験による評価では、提案手法であるPolicy Gradient Stock GAN (PGSGAN) と Policy Gradient Stock GAN with Hinge Loss (PGSGAN-HL) が、次の注文生成において、先行研究と比べてどの程度の性能なのかを検証した。実験データとしては、東京証券取引所の実データを用いて、各種GANの学習を実施した。その後、偽注文の分布と実注文の分布をMSEとKLDで比較することで、それぞれのモデルの性能を評価した。結果、提案手法が先行研究を上回る性能を達成していることが確認できた。加えて、Policy Gradientを導入した副産物として、生成されたPolicyのエントロピーが学習の進み具合を確認するのに有用であることも確認できた。さらに、提案手法において、Hinge Lossを導入したバージョン (PGSGAN-HL) が、学習途中における勾配消失を解消できるために、よりよい性能を達成することが多いことも確認できた。以降の章では、基本的にこの章で構築した実データに基づくGANを用いてさらなる手法の提案と実験を行う。

第4章 GANによるシミュレーション評価

4.1 シミュレーションの評価の難しさと研究背景

社会シミュレーションは様々な社会問題解決に有用である一方で、シミュレーション自体の妥当性の評価はとても難しい。一般に、社会シミュレーションは、あいまいな社会現象を取り扱うのが一般であるため、定性的な評価が中心であり、定量的な評価が困難である。例えば、感染症の拡大の分析を目的として社会シミュレーションを実施する場合、正確な患者の数を予測しようとするることはナンセンスであろう。一方で、大まかな傾向性を予測することはとても重要であると思われる。現在の感染症の拡大速度が指数的な爆発的感染拡大であるかどうかなどはとても重要な定性的な事項である。別の例として、人工市場（金融市場）シミュレーションの場合、市場を支配する支配方程式が存在しないため、市場の真の正しい動きというものを定義することはできない。その代わりに、Stylized Factsと呼ばれる、実際の市場でよく知られる定性的な現象を確認することで、定性的な妥当性の評価を行うことができる。

しかしながら、定性的評価には様々な限界が存在することから、定量的な評価は難しいものの、取り組まなければならぬと考える。例えば、複数のシミュレーションモデルが存在するときに、そのどのモデルが、現実性の観点でよいのかどうか、という比較に関しては、定量的な指標がないと比較が難しい。加えて、シミュレーションモデルのパラメータの調整も、定量的な指標がないと難しい。多くのマルチエージェントシミュレーションにおいては、人間が実際の意思決定や行動プロセスを考慮してモデル構築を行っているため、非常に多くのモデルの内部パラメータが事前に決定できず、後から実際の現象を再現できるかという点でパラメータ決定が行われている。一方で、このパラメータの存在とパラメータチューニングはマルチエージェントシミュレーションにおいては不可欠なものとなってしまっており、ここに対して、定量的な評価を通じたアプローチが実現できるメリットは大きい。特に、定量的な評価が可能となれば、パラメータチューニングの自動化への道も開けてくる。

実際のパラメータチューニングの例を見てみると、Torii ら [53] は、人工市場シミュレーションのパラメータを決めることができず、最終的に、それぞれのパラメータをグリッドに区切り、すべてのパラメータの組み合わせに対して実験を行うことで、その妥当性を担保した。シミュレーションの評価の KPI を定めて最適化することでも、部分的にはパラ

メータ空間の絞り込みは達成できるものの、完全な現実的なシミュレーションを実現できるパラメータの確定は困難であろう。そのため、定量的な評価指標を作成することで従来の Stylized Facts に基づく定性的評価を置き換えることが、人工市場シミュレーションの今後の発展には必要であると考えられる。

本章においては、この人工市場シミュレーションの定量評価という観点に関して、前章で作成した GAN を用いる手法を提案し、従来の定性指標との整合性の確認を通じてその妥当性に関して検討を行う。

4.2 手法: GAN Evaluator

この節では、主にコア技術について説明する。実験向けの詳細な実装については次節で説明する。

すでに前節で説明した通り、本章では、人間によるシミュレーションの定性的な評価を、GAN を用いることで定量的な評価に置き換えることを目指している。GAN は、その学習機構ゆえに、実データのあいまいな分布特性をつかむことができると考えており、それゆえに、シミュレーションの評価にも使用可能ではないか、と考えている。そのため、提案手法としては、GAN の Critic をシミュレーションの評価装置として使う方法を考える。

図 4.1 は、GAN Evaluator の概要を示している。まず、一般の GAN と同様に GAN の敵対的学习を行う。Generator は、乱数を受け取り、偽データを生成し、Critic は偽データと実データを判別することで敵対的な学習を実現できる。この段階は、一般的な GAN と同じであるため、GAN のモデルやデータに制限は特がない。学習が完了後、Critic のみをシミュレーションの評価に使用する。学習に使用したデータと同じ形式のデータをシミュレーションに基づいて作成する。このシミュレーションのデータを Critic に投入することで、Critic はこのデータが偽物か本物かを判定する。この Critic の出力値をシミュレーションの評価値とみなす。例えば、GAN のモデルとして、初期の GAN [86] などのクラス分類ベースの GAN を採用した場合には、シミュレーションの評価は尤度として得ることができる。一方で、WGAN [103] のような距離ベースのモデルを採用した場合には、シミュレーションの評価はスカラー値として獲得することができる。本手法における唯一の制約は、実データのフォーマットとシミュレーションデータのフォーマットが同じであることである。

4.3 実験用モデルの実装

この節では、実験に当たって、前述のコア技術を実際に実装したモデルの詳細について説明する。本研究では、金融市場にフォーカスしたモデルを実装した。GAN のモデルとしては、3 章で作成をした GAN を活用する。一方で、マルチエージェントシミュレーションとしては、先行研究で使用されている人工市場モデルを採用する。その詳細を以下で個別に説明する。

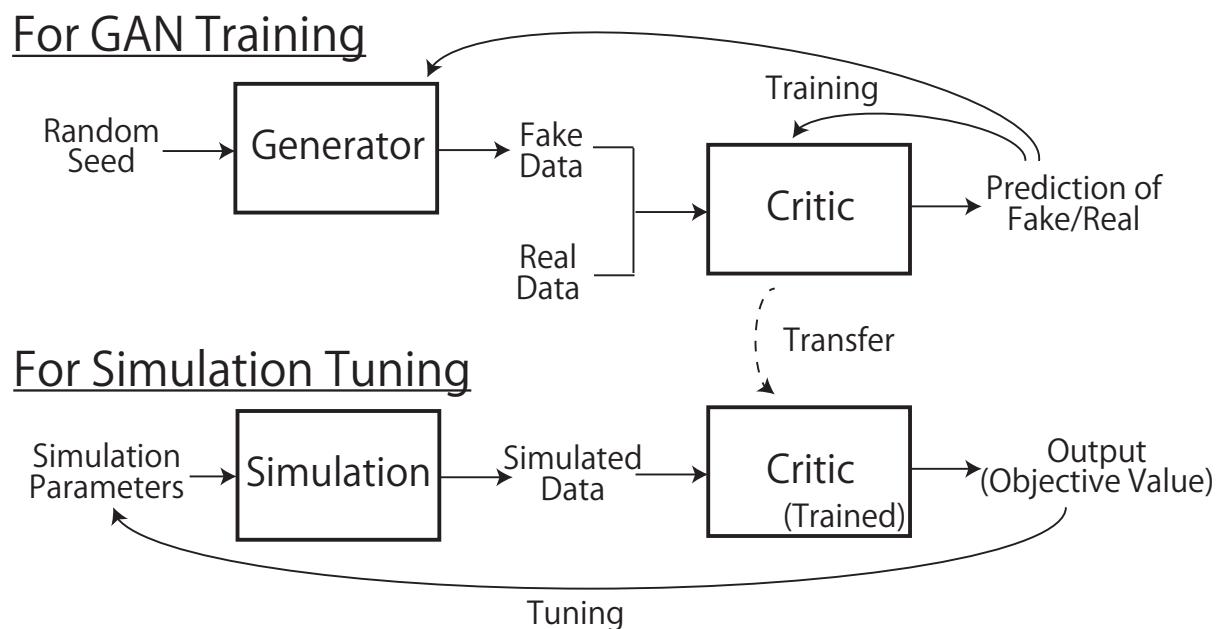


図 4.1: GAN Evaluator の概要. 一般の GAN と同様に GAN の敵対的学习を行う (上段). Generator は、乱数を受け取り、偽データを生成し、Critic は偽データと実データを判別することで敵対的な学习を実現できる. 学习が完了後、Criticのみをシミュレーションの評価に使用する (下段). 学习に使用したデータと同じ形式のデータをシミュレーションに基づいて作成を行う. このシミュレーションのデータを Critic に投入することで、Critic はこのデータが偽物か本物かを判定する. この Critic の出力値をシミュレーションの評価値とみなす.

4.3.1 実験で使用したGANモデル (PGSGAN)

前述の通り、3章で作成したGANである。ここでは、その機構に関して、簡易版を再掲する(図4.2)。

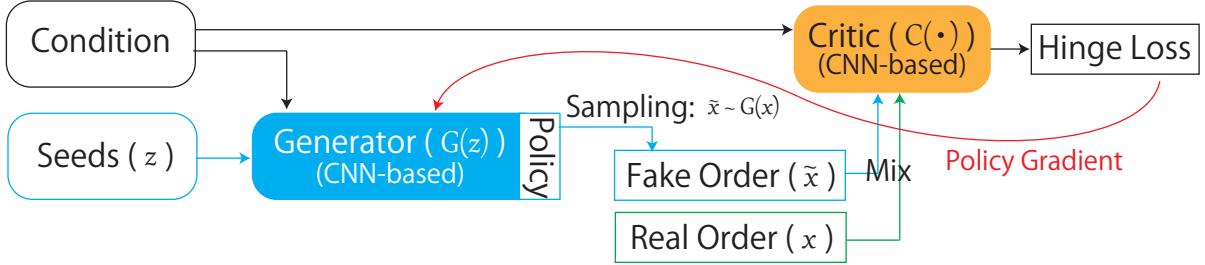


図4.2: PGSGANの概略(簡易版)

なお、本章での実験では、Hinge loss [139]をCriticの最終層に追加したPGSGAN-HLを採用した。前章で記載した通り、Hinge Lossは以下のように定義される：

$$\begin{cases} \max(x + 1, 0) & (\text{critic training for fake data}) \\ \max(1 - x, 0) & (\text{critic training for real data}) \\ x & (\text{generator training}) \end{cases} \quad (4.1)$$

このHinge Lossは学習時のみに挿入されるので、学習時はCriticの出力が $[-1, 1]$ に制限される。一方で、学習時ではない時は、Hinge Lossが使用されない。しかしながら、学習時に、出力が $[-1, 1]$ から外れる場合は、そこで勾配が切られてしまうために、 $[-1, 1]$ を超えるような出力は極めて発生しにくい。これらを踏まえると、PGSGAN-HLの出力が-1に近ければ偽注文、1に近ければ現実的な注文とみなすことができる。この判定をシミュレーションデータに対して行うことで、-1から1の範囲でシミュレーションの現実性を評価することができる。

Criticをシミュレーションの評価に使用する際には、シミュレーションデータを毎ステップごとに入力をして得られるすべての出力の平均をとる。例えば、100個の時系列データがシミュレーションにより生成される場合、PGSGAN-HLでは20のヒストリカルデータを条件データとして用いるため、21番目から評価が開始され、全部で80個のデータポイントに対して評価が行われる。これらを平均することで評価値を計算できる。

4.3.2 実験で使用したシミュレーションモデル

本研究では、先行研究[53]で用いられている、一般的な人工市場モデルを採用した。このシミュレーションはPlhamJ上で使用可能であり、PlhamJとは、A Platform for Large-scale, and High-frequency Artificial Financial Markets (Java version) [58]である。以下では、そのモデルについて詳細に説明する。

まず、シミュレーションには n_{agent} 体の Stylized Trader エージェントと 1 つの連続ダブルオーファクション形式の市場が存在する。時刻 t において、Stylized Trader エージェント i は、ファンダメンタル、チャート(トレンド)、ノイズファクターに基づいて、注文行動の決定を行う。まず、エージェントは下記の通り 3 ファクターを計算する。

- ファンダメンタルファクター：

$$F_t^i = \frac{1}{\tau^{*i}} \ln \left\{ \frac{p_t^*}{p_t} \right\}. \quad (4.2)$$

ここで、 τ^{*i} はエージェント i の平均回帰時間のパラメータであり、 p_t^* は時刻 t におけるファンダメンタル価格、 p_t は時刻 t における価格である。

- チャート(トレンド)ファクター：

$$C_t^i = \frac{1}{\tau^i} \sum_{j=1}^{\tau^i} r_{(t-j)} = \frac{1}{\tau^i} \sum_{j=1}^{\tau^i} \ln \frac{p_{(t-j)}}{p_{(t-j-1)}}. \quad (4.3)$$

ここで、 τ^i はエージェント i の Time Window Size であり、 r_t は時刻 t におけるログリターンであり、価格時系列から計算可能である。

- ノイズファクター：

$$N_t^i \sim \mathcal{N}(0, \sigma). \quad (4.4)$$

ここで、 N_t^i は、平均 0、分散 $(\sigma)^2$ の正規分布を意味する。

続いて、エージェントは、上記の 3 つのファクターの重み付き平均を下記のように計算する：

$$\hat{r}_t^i = \frac{1}{w_F^i + w_C^i + w_N^i} (w_F^i F_t^i + w_C^i C_t^i + w_N^i N_t^i). \quad (4.5)$$

ここで、 w_F^i, w_C^i, w_N^i は、エージェント i の 3 つのファクターに対する重み付けである。

続いて、エージェント i の期待価格が以下の通り計算される。

$$\hat{p}_t^i = p_t \exp \left(\hat{r}_t^i \tau^i \right). \quad (4.6)$$

そのうえで、固定注文マージン $k^i \in [k_{\min}, k_{\max}]$ を用いて、最終的な注文は以下のように計算される。

- もし、 $\hat{p}_t^i > p_t$ であれば、エージェント i は以下の価格で買い注文を立てる。

$$\left[\min \left\{ \hat{p}_t^i (1 - k^i), p_t^{\text{ask}} \right\} \right]. \quad (4.7)$$

- もし、 $\hat{p}_t^i < p_t$ であれば、エージェント i は以下の価格で売り注文を立てる。

$$\left[\max \left\{ \hat{p}_t^i (1 + k^i), p_t^{\text{bid}} \right\} \right]. \quad (4.8)$$

ここで, p_t^{bid} と p_t^{ask} は, 売りと買いの裁量気配値を意味する. また, $\lfloor \cdot \rfloor$ は床関数を, $\lceil \cdot \rceil$ は天井関数を意味する. これらの関数は, 呼値による価格の刻みをシミュレーションに反映するために適用している.

固定パラメータとして, $p_t^* = 1000$, $w_N^i \sim Ex(1.0)$ を採用した. 他のパラメータとして, $w_F^i \sim Ex(w_F)$ と $w_C^i \sim Ex(w_C)$ の代表値である $w_F, w_C, \sigma, \tau^* \in [\tau_{\min}^*, \tau_{\max}^*]$, $\tau \in [\tau_{\min}, \tau_{\max}]$, $k^i \in [k_{\min}, k_{\max}]$ が存在し, パラメータチューニングにより決定される. ここで, $Ex(\lambda)$ は, 平均と分散が λ である指数分布を意味する.

4.4 実験と評価

実験と評価は, 大きく分けて 3 ステップで実施した. 以下ではそれぞれのステップについて説明する.

4.4.1 実験: パラメータチューニング

第一ステップとして, GAN Evaluator を用いた人工市場シミュレーションのパラメータチューニングを実施する.

本研究の目的は, 従来の定性的な評価を定量的な評価に置き換えることである. そのため, パラメータチューニングの手法そのものをここでは提案しているわけではないので, デファクトスタンダードなベイズ推定に基づいたパラメータチューニングのアルゴリズムを採用することにした. チューニングアルゴリズムとしては, tree-structured Parzen estimator (TPE) [65] を採用した. このチューニングアルゴリズムは, 機械学習のパラメータチューニングにおいて比較的よく使用されており, Optuna [66] として知られている. TPE はベイズ推定による手法であり, ブラックボックス最適化として, 目的関数を最大化するようにパラメータの探索を行う. このパラメータチューニングを通じて, 適切なチューニングができるることを確認することにより, GAN Evaluator の手法の妥当性を確認する.

すでに 4.3.2 節で述べた通り, 一部のパラメータのみをチューニングの対象としている. また, ベイズ推定のアルゴリズムを採用したため, ここでは, 事前にそれぞれのパラメータのチューニングのレンジを設定する. 最適解がどこに存在するのかは自明ではないので, 先行研究 [53] を参考にしながら, 比較的広くパラメータのレンジを決めることにした. (先行研究 [53] では, 最終的なパラメータしか記載されていないが, それを参考にレンジを決めた.) 以下が本研究で採用したレンジである.

- エージェント数: $n_{\text{agent}} = 2, 3, \dots, 1000$
- ファンダメンタルファクターの重み: $w_F \in [0, 50]$
- チャート (トレンド) ファクターの重み: $w_C \in [0, 50]$

- ノイズスケール: $\sigma \in [10^{-10}, 10^{-1}]$ (対数スケールでチューニング)
- 平均回帰時間定数の最小値: $\tau_{\min}^* \in [1, 10^5]$ (対数スケールでチューニング)
- 平均回帰時間定数の最大値: $\tau_{\max}^* \in [\tau_{\min}^*, 10^5]$ (対数スケールでチューニング)
- Time Window Size の最小値: $\tau_{\min} \in [1, 10^5]$ (対数スケールでチューニング)
- Time Window Size の最大値: $\tau_{\max} \in [\tau_{\min}, 10^5]$ (対数スケールでチューニング)
- 固定注文マージンの最小値: $k_{\min} \in [0, 1]$
- 固定注文マージンの最大値: $k_{\max} \in [k_{\min}, 1]$

各実験においては、1000ステップのプレオープニング・セッション(ザラ場前の注文可能時間)と100,000ステップからなるザラ場のセッションからなるシミュレーションを行う。そして、そのシミュレーションの出力をGANのCriticに入力として与えることで得られる出力が最大化されるようにチューニングを行う。

しかしながら、パラメータ数が多い場合、そのチューニングを同時に行うのは若干難しい。例えば、目的関数に無相関なパラメータが存在すると、チューニングの精度に影響が出る可能性や、パラメータ確定に難航する可能性が考えられる。そのため、複数ステップに分けたチューニングを行う。複数ステップでのチューニングの各ステップでは、10回のチューニング試行の結果の分散が中央値の10%以下になったものから固定していくということを繰り返す。最終的に、固定できるパラメータがなくなったところで、複数ステップでのチューニングは完了とみなし、残りのパラメータを中央値で固定することとした。

なお、それぞれのステップのチューニングでは、1,000回の試行(以下、チューニングステップと呼ぶ)を通じてチューニングを行う。加えて、3章で10銘柄のGANを作成したので、この10銘柄に対するGANのそれぞれをGAN Evaluatorとして採用した場合の出力値の平均をとることとした。加えて、各試行においては、シミュレーションの結果を安定させるため、10回のシミュレーションを実施した。

つまり、これらをまとめると、10回のシミュレーションの結果を10個のGANモデルにかけることで、100個の出力が獲得できる。この平均を1つのパラメータセットに対する目的関数の値としてみなす。この目的関数をチューニングする操作を1000回行うことを10回行い、その中央値と分散に基づいて固定するパラメータを確定するというのが複数ステップチューニングの1ステップであり、これを繰り返す。そのため、複数ステップチューニングの1ステップでは、 $10 \text{ simulations/tuning step} \times 1000 \text{ tuning steps/trials} \times 10 \text{ trials} = 100000 \text{ simulations}$ が必要である。実際の実験では、複数ステップチューニングは3ステップで完了することができたので、最終的にこのチューニングには300000シミュレーションを要するということである。

チューニングの完了後、Stylized Factsが再現できているかの検証を行う。Stylized Factsというのは、金融市場において特有的に観測できる統計的性質であり、一般に、市場の設

置国や種類に寄らず、広く確認できることの多いものである。このStylized Factsは、金融市場の特性を把握するために、実証的に研究がされており、その実証研究[82]に基づいたものを本研究でも採用する。一方で、Stylized Factsは、クロスセクショナル分析などの、単一株の時系列では検証できない特性も多く含まれており、本研究のシミュレーションで確認できるものには限りがある。そのため、本研究では、先行研究[82]に基づき、本研究でも検証可能であろうものを選択的に検証を行うこととした。実際に検証が行えるであろうと判断したStylized Factsは以下のとおりである。

- Absence of Autocorrelations：株価のリターン時系列には自己相関性が極めて低くなっている、遅くとも数Tick程度のラグまでで自己相関が消失する。
- Heavy Tails Property：株価のリターン分布がガウス分布に比べて、分布のテイル部分の確率が高くなる。金融市場では、 2σ (σ は標準偏差)を超える価格変動が頻発するという現象として理解できる。
- Aggregational Gaussianity：株式市場においてみられるHeavy Tails Propertyのような現象は、測定のラグを長くすることにより、標準ガウス分布に近づいていく。つまり、長期的なデータ分析の方が正規性が高まる。
- Volatility Clustering：金融市場において、ボラティリティは時間方向に相関を持つ。これは、ボラティリティの大きい時間帯と小さい時間帯に分かれていることが多いという現象として理解できる。

これらの観測に加えて、各時間ラグごとの対数リターンの尖度と歪度をプロットすることで、さらなる検証も行うこととした。

これらのStylized Factsの再現性の評価は難しいため、下記の3段階評価で評価を行う。

- +: 適切に観測される
- -: 観測されない。ただし、人によって判断が変わる可能性が否定できない程度にはあいまいである。
- --: 適切に観測されない。かなり客観的な根拠がある。

しかしながら、この判断結果のみを提示するのは、いささか情報不足であると考えられる。そのため、判断の詳細は後続の評価2の段階で示す。

4.4.2 評価1：チューニング済みパラメータのアブレーションテスト

この評価ステップにおいては、前節の実験でチューニングされたパラメータから1つだけパラメータを変更したシミュレーションパラメータセットで実験を行った場合の変化を比較する。全部で10個のパラメータがあるので、それぞれに対して実施することし、可

能な限り極端なパラメータの変更を行うこととした。ただし、パラメータの変化は前述のチューニングに使用したレンジ内から選ぶこととする。この変化させたパラメータセットの結果に対して、同様に3段階評価でStylized Factsを検証する。

この評価の目的は大きく分けて2つ存在する。

- このチューニングタスクの難易度の確認
- 伝統的な定性的評価と提案手法である定量評価手法の対応性を確認するためのサンプル作成

まず、1つ目の点についてだが、そもそも今回取り組んでいるパラメータチューニングが非常に簡単な者でないことを確認するためである。もし、今回のチューニングが非常に簡単で、最適値のエリアが広かったとすると、GAN Evaluatorの妥当性の担保ができない。一方で、たった一つのパラメータを変化させた程度でもStylized Factsが確認できないような難易度であるということが確認できれば、GAN Evaluatorがその限られた最適値に誘導するのに十分な性能を持つことを示せることになる。加えて、2点目についても、GAN Evaluatorの妥当性を確認するために非常に重要である。GAN Evaluatorの妥当性の1つとして、従来のStylized Factsとの対応が取れていることがあげられる。金融市場においては、Stylized Factsの再現という点が重要であるとともに、今回の定量化の目的の一つにシミュレーション間の比較があげられるため、Stylized Factsがどの程度再現されているかという順位的な比較との整合性も重要である。この確認を行うためには、複数のパラメータセットに対して評価を行い、それらを比較する必要がある。一方で、パラメータの変化が微小であると、その微小な変化の結果として得られる出力の変化も微小となり、Stylized Factsの再現性における順位的比較が困難になるため、比較的大きなパラメータ変化をさせて、それらを比較するのが妥当である。

この実験では、一つだけのパラメータを悪い方向に変化させて実験するため、アブレーションテストと呼ぶこととする。また、それぞれのアブレーションテストのパラメータセットに対しては、1000回のシミュレーションを行い、それを集計して評価を行う。

4.4.3 評価2：Stylized Factsの詳細分析

前述の通り、単に3段階評価の結果のみを示すだけでは説得力がないため、プロットを示すことで、詳細な分析の確認を行う。まずは、今回の実験において、キーとなる部分について、プロットの比較を行い、その後、すべての結果に関して提示をする。

4.5 結果

4.5.1 実験：パラメータチューニング

3ステップのパラメータチューニングの結果、下記の通りのパラメータを獲得した。

- エージェント数： $n_{\text{agent}} = 929$
- ファンダメンタルファクターの重み： $w_F = 36.6$
- チャート(トレンド)ファクターの重み： $w_C = 2.29$
- ノイズスケール： $\sigma = 5.52 \times 10^{-6}$
- 平均回帰時間定数の最小値： $\tau_{\min}^* = 669$
- 平均回帰時間定数の最大値： $\tau_{\max}^* = 64700$
- Time Window Size の最小値： $\tau_{\min} = 15.1$
- Time Window Size の最大値： $\tau_{\max} = 10470$
- 固定注文マージンの最小値： $k_{\min} = 0.0314$
- 固定注文マージンの最大値： $k_{\max} = 0.0552$

これらのチューニング後のパラメータを使用した場合の GAN の出力は 0.96524 となった。

これらのパラメータを用いた場合は、Stylized Facts は適切に観測された。その詳細については 4.5.3 章で示す。Volatility Clustering のみに関しては、完全に明確には観測できなかったものの、Realized Volatility の自己相関は短時間または中程度のスパンにおいては、明確に正の傾向が確認できたので、問題ないものと見做している。また、先行研究 [82] によれば、この自己相関は、実市場においては正ではあるものの、かなり小さい値(0.1 以下程度)であることがわかっており、これと整合性の取れる結果であることも、問題がないと判断できる根拠であると考えている。詳細については、4.5.3 節を参照されたい。

4.5.2 評価 1：チューニング済みパラメータのアブレーションテスト

表 4.1: アブレーショントレーストの結果. 各行は GAN Evaluator の出力の順で並んでいます. Stylized Facts と書かれた列に関しては, それ以降の個別の Stylized Facts の観測結果をまとめたものである. ACF, Agg. Gaus., Vol. Clus., Kurt./Skew. はそぞれぞれ, Autocorrelation Function (Absence of Autocorrelation と同義), Aggregational Gaussianity, Volatility Clustering, Kurtosis/Skewness (尖度/歪度) を意味する. 塗りつぶされたセルに関しては, 4.5.3 節で特に重点的に取り扱う.

#case	Param.	Value Change	GAN	Stylized Facts		ACF	Heavy Tail	Agg. Gaus.	Vol. Clus.	Kurt./Skew.
				ACF	Heavy Tail					
0	-	-	0.96524	+	+	+	+	+	+	+
1	τ_{\min}	$15.1 \rightarrow 1$	0.96523	+	+	+	+	+	+	+
2	τ_{\max}^*	$64700 \rightarrow 669$	0.95020	+	+	+	+	+	+	+
3	k_{\min}	$0.0314 \rightarrow 0.0552$	0.91626	+	+	+	+	+	+	+
4	τ_{\min}^*	$669 \rightarrow 0$	0.91145	+	+	+	+	+	+	+
5	w_C	$2.29 \rightarrow 50$	0.86420	+	+	+	+	+	+	+
6	n_{agent}	$929 \rightarrow 20$	0.84225	-	+	+	+	-	+	+
7	k_{\max}	$0.0552 \rightarrow 1$	0.68891	-	+	+	-	-	-	+
8	n_{agent}	$929 \rightarrow 2$	0.64020	--	+	+	No price movement occurs (-)	-	-	-
9	k_{\min}	$0.0314 \rightarrow 0$	0.51658	-	+	+	-	-	-	-
10	τ_{\max}	$10470 \rightarrow 15.1$	0.31392	--	--	+	-	+	-	-
11	w_F	$36.6 \rightarrow 0$	0.02262	--	+	+	-	-	-	-
12	σ	$5.52 \times 10^{-6} \rightarrow 0.1$	-1	--	--	Price goes infinity (-)				

表4.1は、アブレーションテストの結果を示している。各行は、変化させたパラメータとパラメータの変化値について掲載している。一番上の行がパラメータの変化をおこなっていない最も良い状況の結果を示しており、これは前述の最適なパラメータを採用した場合の結果と同一である。また、既に述べた通り、パラメータ変化はできるだけ極端におこなっている。パラメータを変化させるにあたって、極端に大きくしても小さくしても変化幅に大きな違いがない場合は、その双方で実験をおこなっている。この表では、GANの出力値の大きい順(第4列)に基づいてソートをしている。

Stylized Factsと記載された第5列は、前述の3段階評価のそれぞれのStylized Factsの結果を総合した結果である。この結果をGANの出力値と比較してみると、従来の定性的なStylized Factsに基づく評価と、GANの出力値の間に、それなりの順位相関があることがわかる。また、この総合評価の詳細の内訳については、第5列目以降に掲載している。次の節では、詳細についてプロットとともに明示する。

4.5.3 評価2：Stylized Factsの詳細分析

本節では、前節で示した結果に関して、その個別の詳細判断をプロットとともに説明する。なお、詳細な判断に関する説明は、表4.1で塗りつぶしたセルに対して重点的に行い、他についてはプロットを示すだけにとどめる。

Absence of Autocorrelation (ACF)

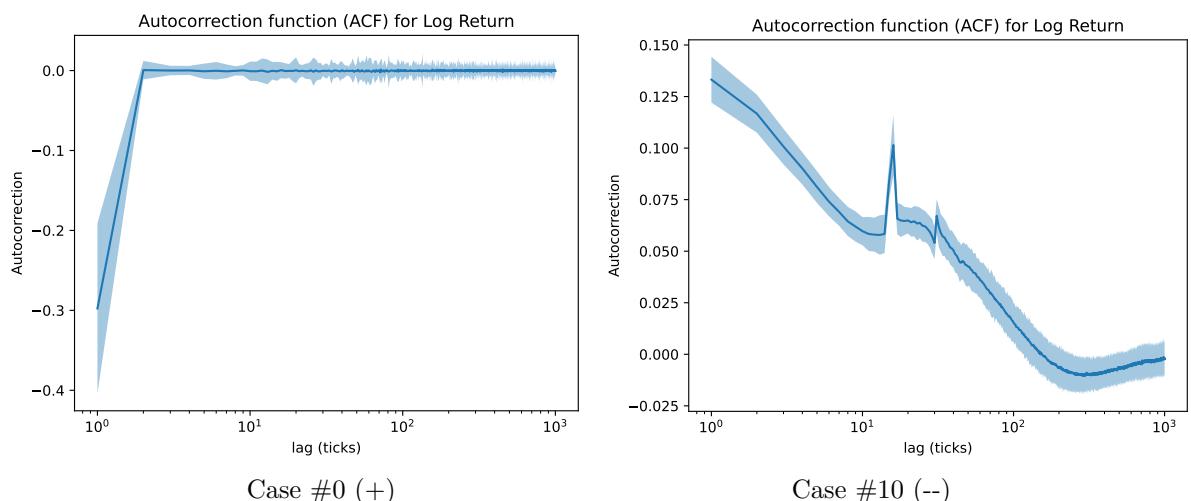


図4.3: 対数リターンのAutocorrelation Function (ACF)。Case #0(最適値)とCase #10の結果。横軸は対数リターンの計算に使用したラグであり、縦軸が自己相関の値である。塗りつぶしエリアは、1000試行の結果から得られる95% Confidence Interval (CI)を示している。

図4.3は、Autocorrelation Function (ACF) 分析のCase #0と#10の結果を示しており、Stylized Factsの一つであるAbsence of Autocorrelationを確認するためのプロットである。このプロットでは、各ラグにおける対数リターン時系列の自己相関性について見ていている。つまり、

$$\text{Autocorrelation}(l) = \text{Corr.} \left(\ln \frac{p_t}{p_{t-1}}, \ln \frac{p_{t-l}}{p_{t-l-1}} \right) \quad (4.9)$$

と計算される。ここで、 l は、自己相関を計算するためのラグ(プロットの横軸に対応)である。図4.3の左の図は、Case #0のものであり、非常に早く自己相関が0に収束しているため、正常にAbsence of Autocorrelationが確認できると言える。一方で、右の図は、Case #10の結果であるが、自己相関が0に収束しておらず、Absence of Autocorrelationが確認できないといえる。先行研究[82]に基づけば、実際の金融市場においては、自己相関性は対数リターンの計算に使用するラグが2-3 Ticks程度でもすぐに0に収束することが示されている。この事実と、Case #0の結果は綺麗に一致している。一方で、#10の結果は、この傾向を確認できない。そのため、Case #10に対しては、3段階評価の中で最も悪い評価であるとした。

以下に、本分析の全てのCaseの結果を掲載する。

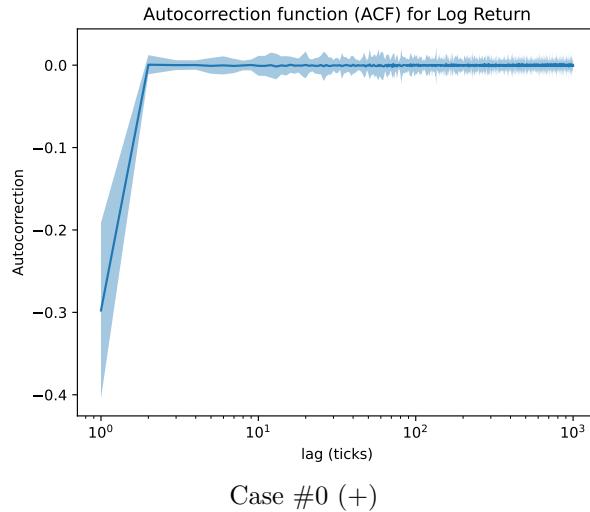


図4.4: 対数リターンのAutocorrelation Function (ACF)。Case #0の結果。横軸は対数リターンの計算に使用したラグであり、縦軸が自己相関の値である。塗りつぶしエリアは、1000試行の結果から得られる95% CIを示している。

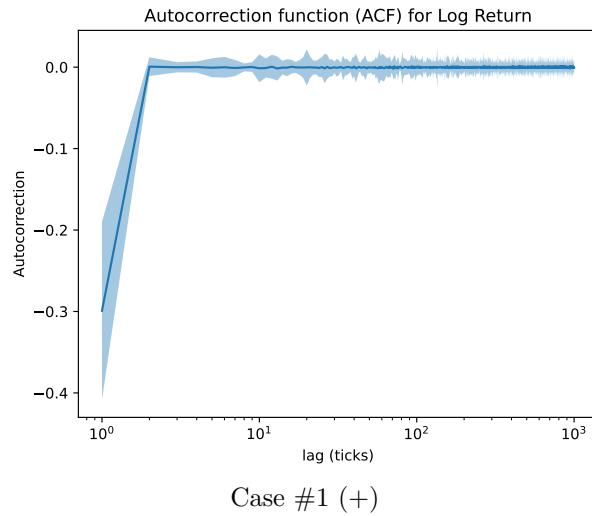


図 4.5: 対数リターンの Autocorrelation Function (ACF). Case #1 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.

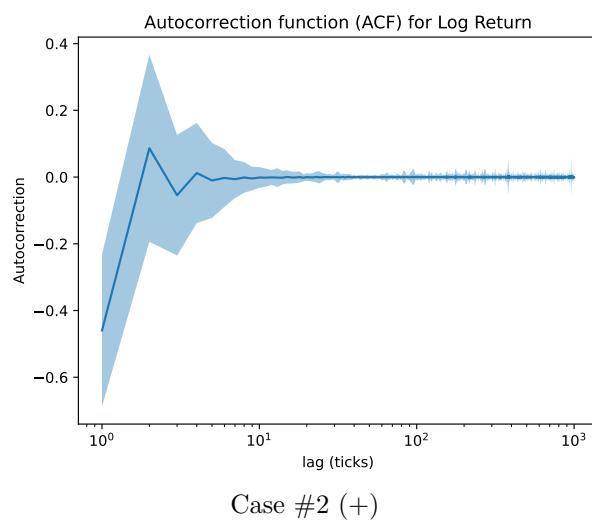
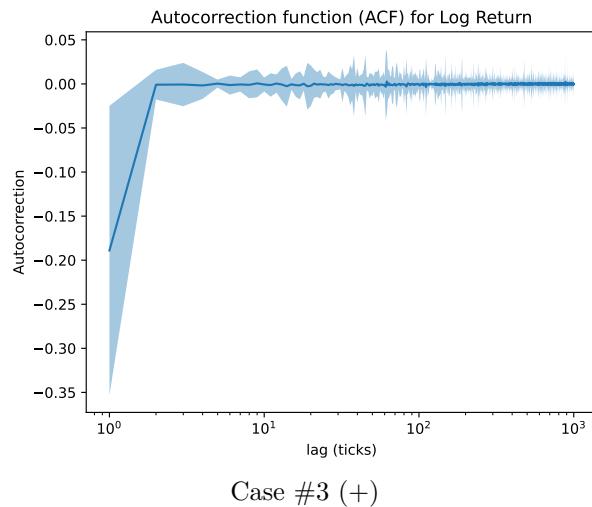
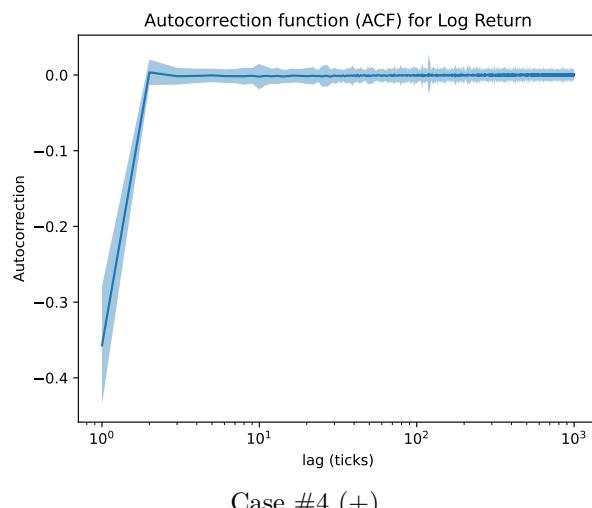


図 4.6: 対数リターンの Autocorrelation Function (ACF). Case #2 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.



Case #3 (+)

図 4.7: 対数リターンの Autocorrelation Function (ACF). Case #3 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.



Case #4 (+)

図 4.8: 対数リターンの Autocorrelation Function (ACF). Case #4 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.

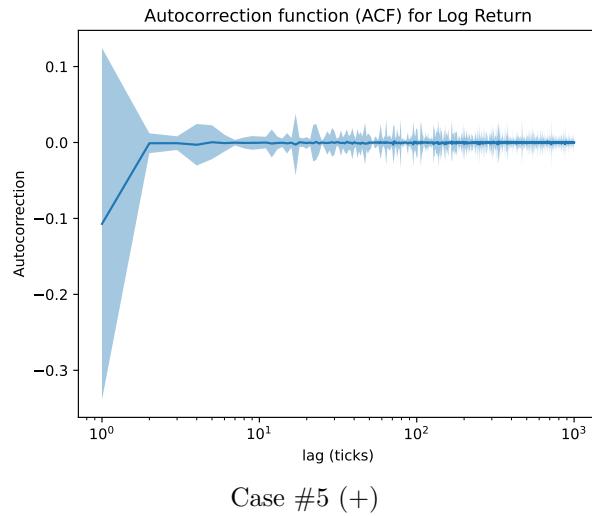


図 4.9: 対数リターンの Autocorrelation Function (ACF). Case #5 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.

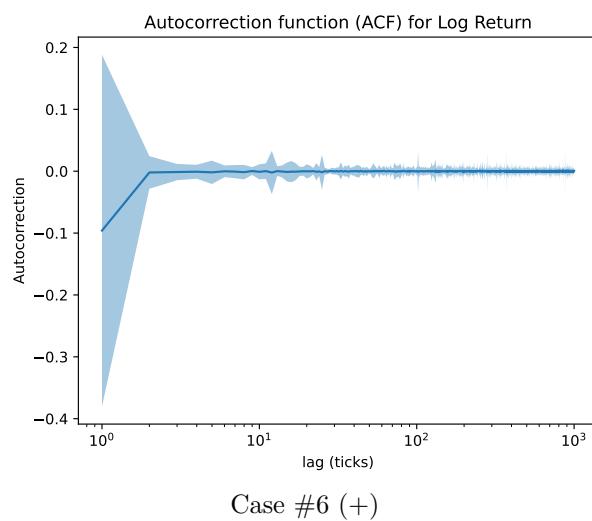
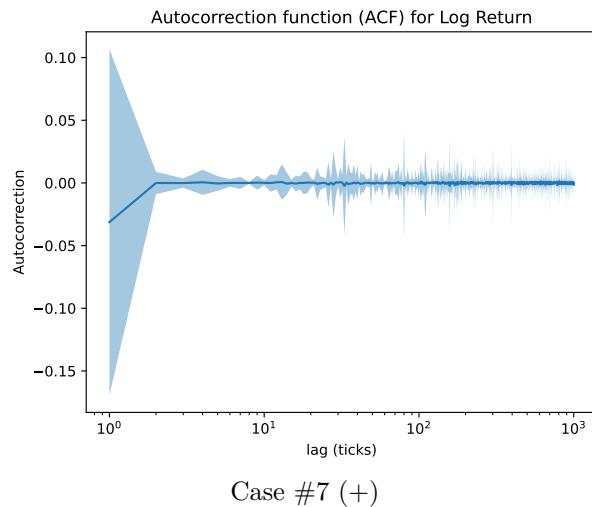
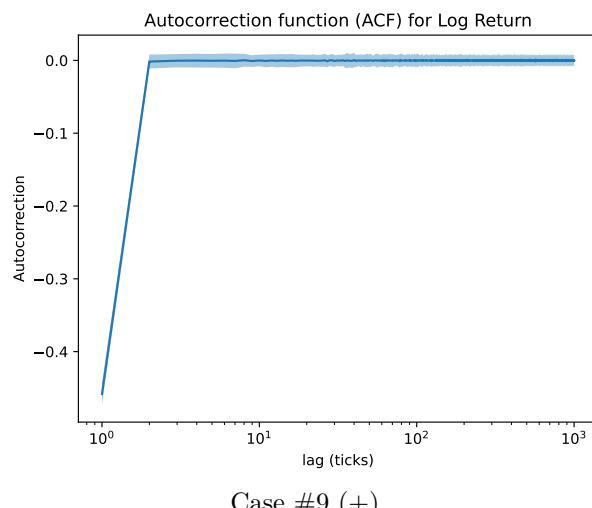


図 4.10: 対数リターンの Autocorrelation Function (ACF). Case #6 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.



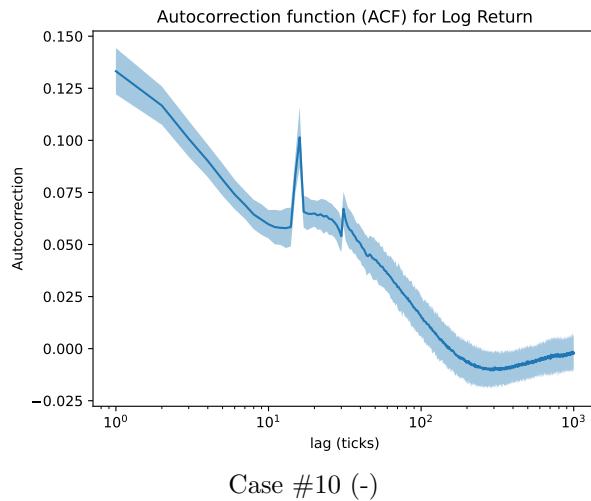
Case #7 (+)

図 4.11: 対数リターンの Autocorrelation Function (ACF). Case #7 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.



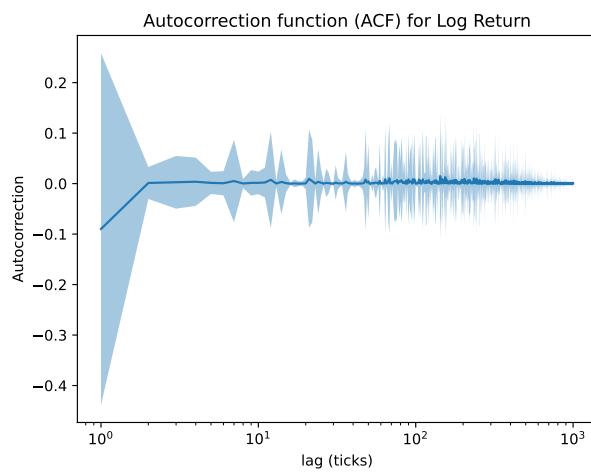
Case #9 (+)

図 4.12: 対数リターンの Autocorrelation Function (ACF). Case #9 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.



Case #10 (-)

図 4.13: 対数リターンの Autocorrelation Function (ACF). Case #10 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.



Case #11 (+)

図 4.14: 対数リターンの Autocorrelation Function (ACF). Case #11 の結果. 横軸は対数リターンの計算に使用したラグであり, 縦軸が自己相関の値である. 塗りつぶしエリアは, 1000 試行の結果から得られる 95% CI を示している.

Heavy Tail & Aggregational Gaussianity

図4.15は、絶対対数リターンの分布を示している。絶対対数リターンは、 $|\ln p_t/p_{t-l}|$ と定義され、 l は計算に使用するタイムラグでそれぞれのプロットで異なっている。この図では、対数リターン分布を絶対値を取る前に正規化してから、絶対値をとっており、横軸が咳聞かされた絶対ログリターンに対応している。オレンジ色の線は、ガウス分布に対応している。そのため、青い線とオレンジ色の線が重なっていれば、シミュレーションの分布のリターンがガウス分布にあっていていることを確認できる。先行研究[82]によれば、この分析により、Heavy Tail PropertyとAggregational Gaussianityを検証できる。Heavy Tailが実際にシミュレーションでも再現されているとすれば、テイル分布(プロットにおける右側)はガウス分布よりも高い確率をもつはずである。一方で、Aggregational Gaussianityに関しては、対数リターンの計算に使用するラグが大きくなればなるほど、対数リターンの分布は正規化を増し、ガウス分布に近くなるとされている。図4.15では、上段と下段左のプロットがCase #0の異なるラグのプロットの結果である。これによれば、Heavy Tail PropertyもAggregational Gaussianityも観測することができる。つまり、ラグが短いプロットにおいては、テイル部分の分布がガウス分布よりも確率が高くなっている上、計算に使用するラグが大きくなればなるほど、ガウス分布に収束している。一方で、右下のプロットは、Case #9のラグが10000の場合の結果であるが、このグラフでは、左下のプロットと異なり、ガウス分布への収束が悪く、異常な波も発生している。一方で、収束が悪いとはいえども、収束していないとまでは断言できない、判断の難しいケースであり、評価が分かれる可能性のあるケースであることから、この結果は“-”と評価した。

以下では、全てのケースでの結果を示す。Heavy Tail Propertyについては全ての結果で観測できたので、ここでは、主にAggregational Gaussianityの判断結果を記載する。

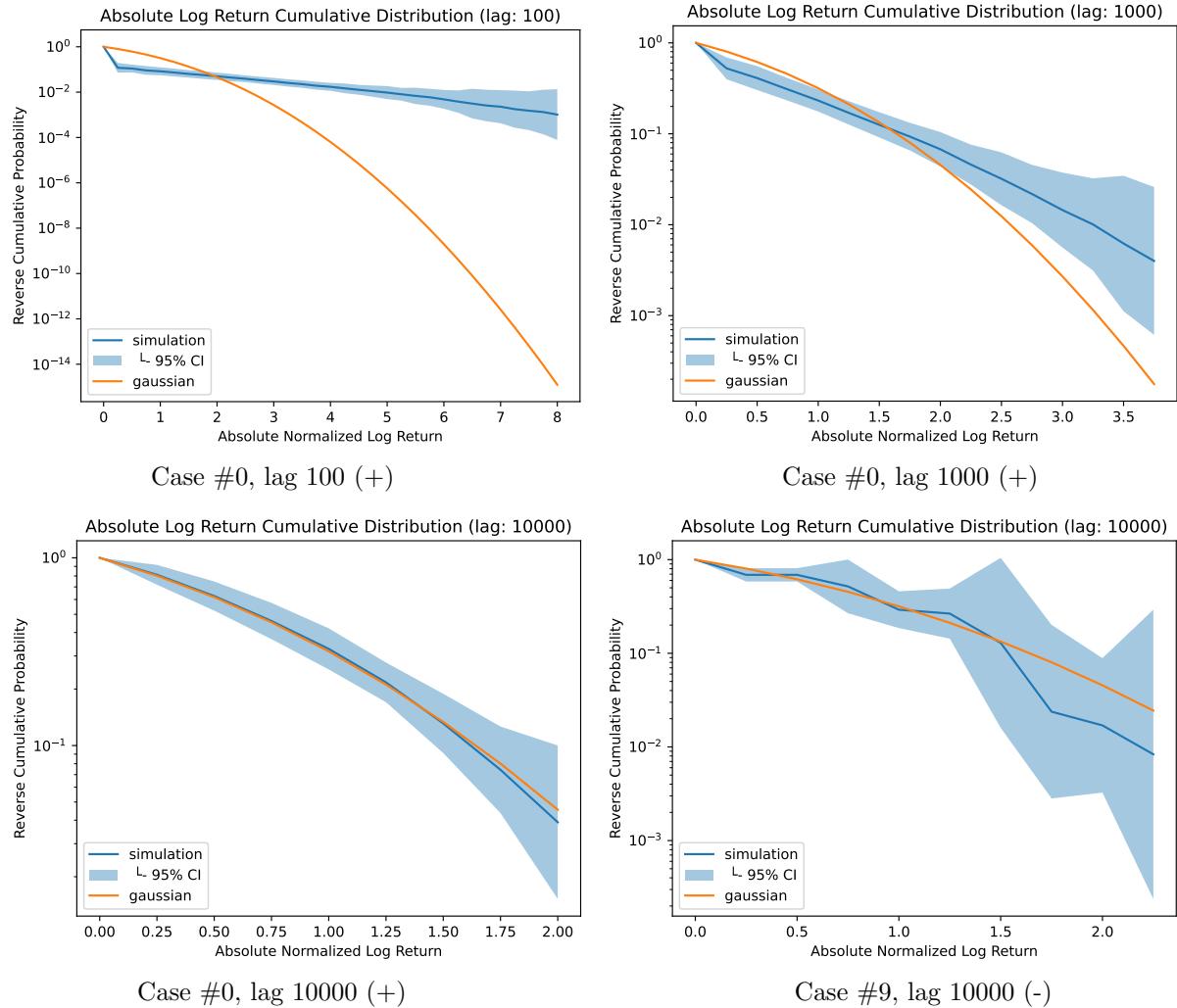


図 4.15: 絶対対数リターンのプロット. 上段と下段の左側は, Case #0 (最適値) の結果. それぞれは, 対数リターンの計算に使用するラグがそれぞれ 100, 1000, 10000 Ticks と異なっている. 右下の図は Case #9 の結果であり, ラグが 10000 の場合である. このケースは, Aggregational Gaussianity の観点で-と判断されている. オレンジ色の線はガウス分布の確率密度関数であり, 塗られているエリアは 1000 試行の結果から算出された 95% CI である.

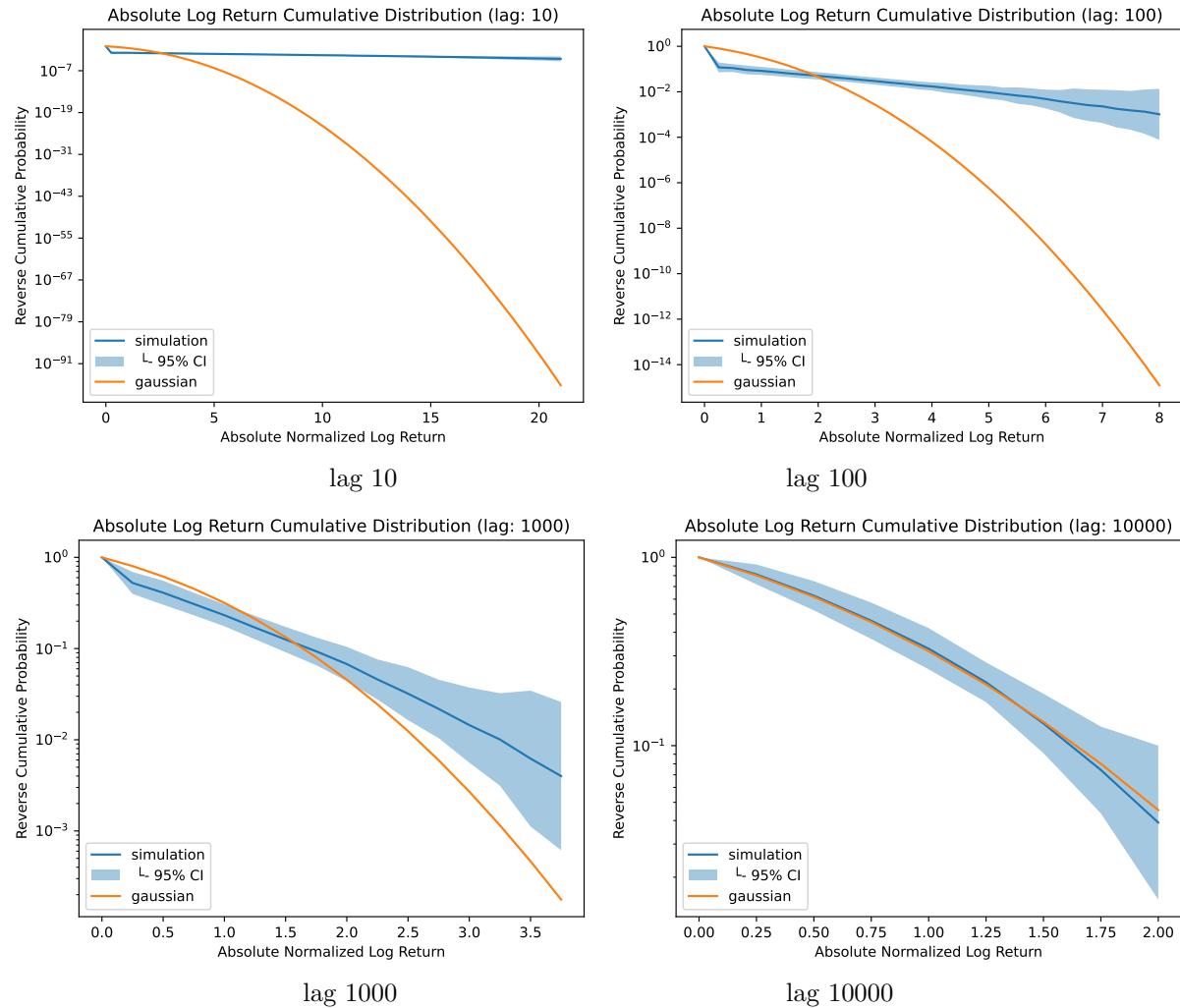


図 4.16: 絶対対数リターンのプロット (Case #0). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は+と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

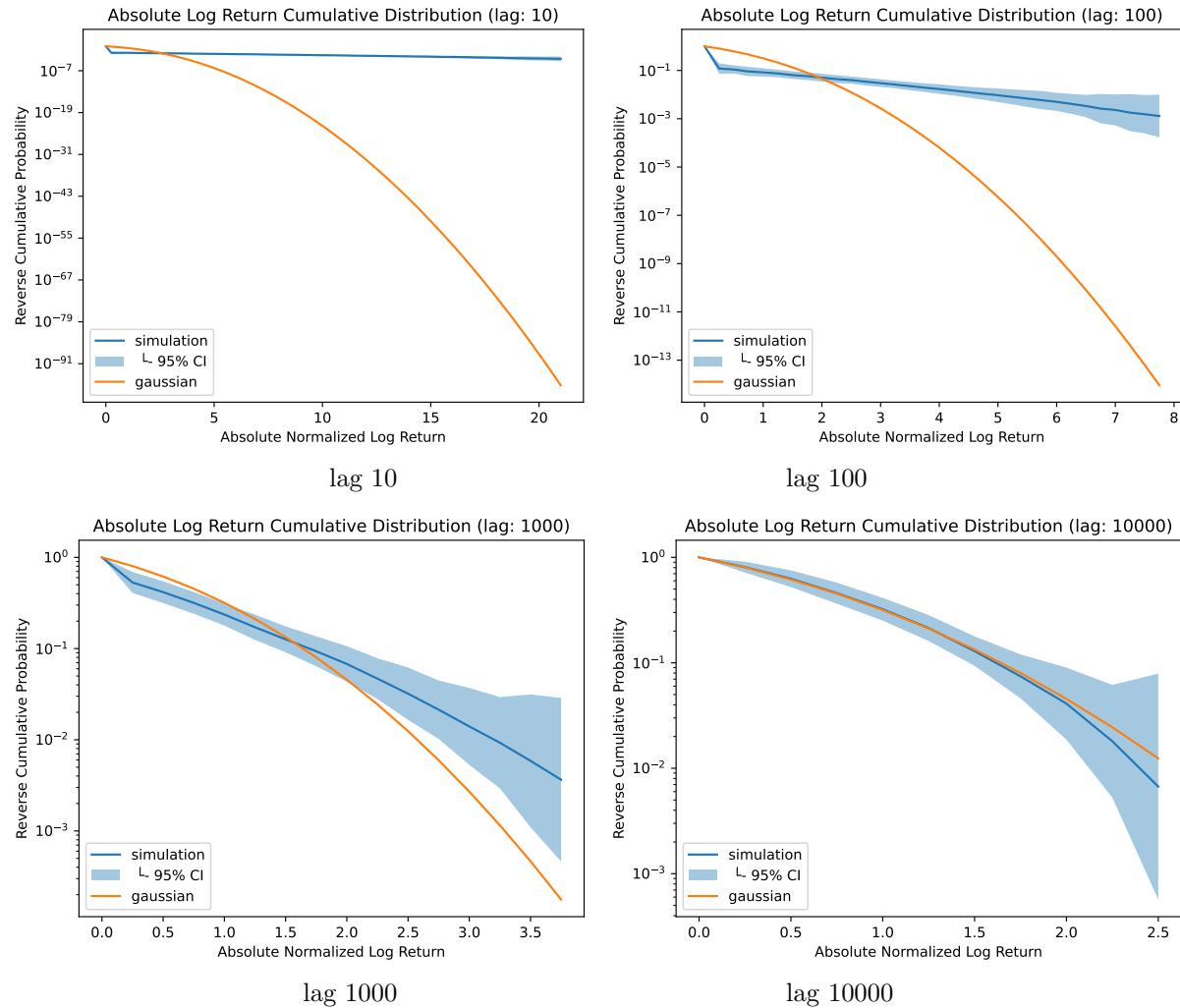


図 4.17: 絶対対数リターンのプロット (Case #1). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は+と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

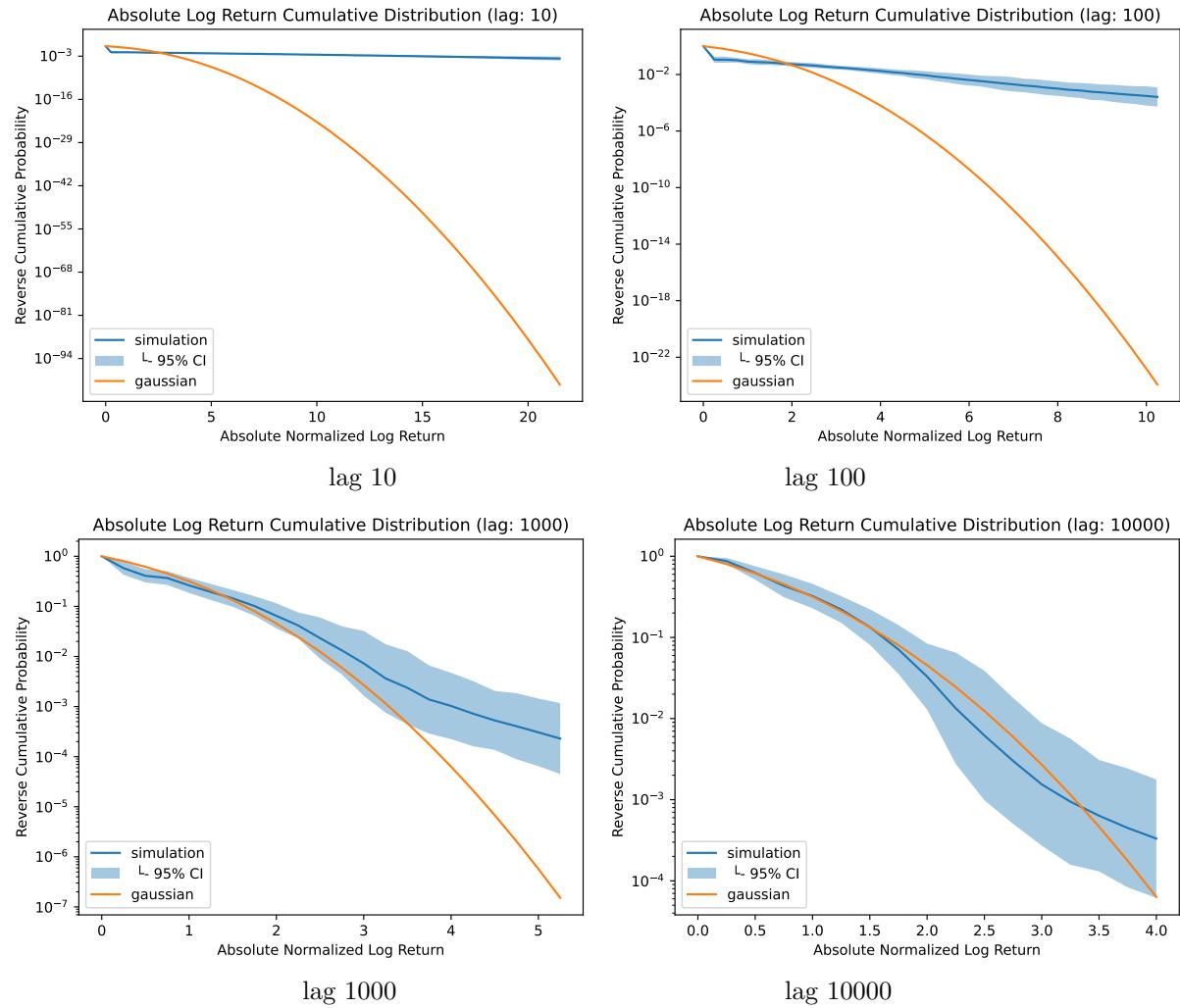


図 4.18: 絶対対数リターンのプロット (Case #2). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は+と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

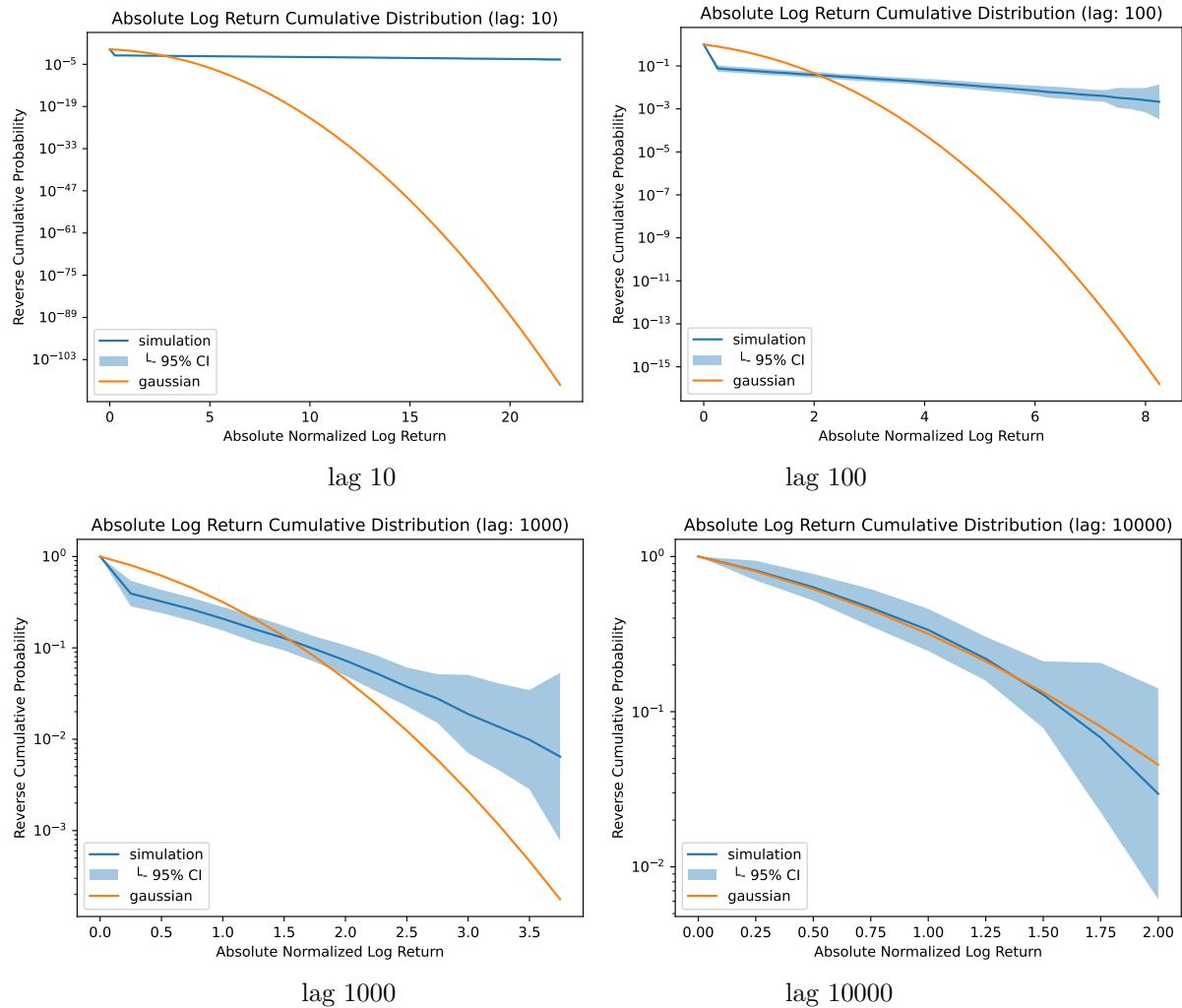


図 4.19: 絶対対数リターンのプロット (Case #3). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+、Aggregational Gaussianity は+と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

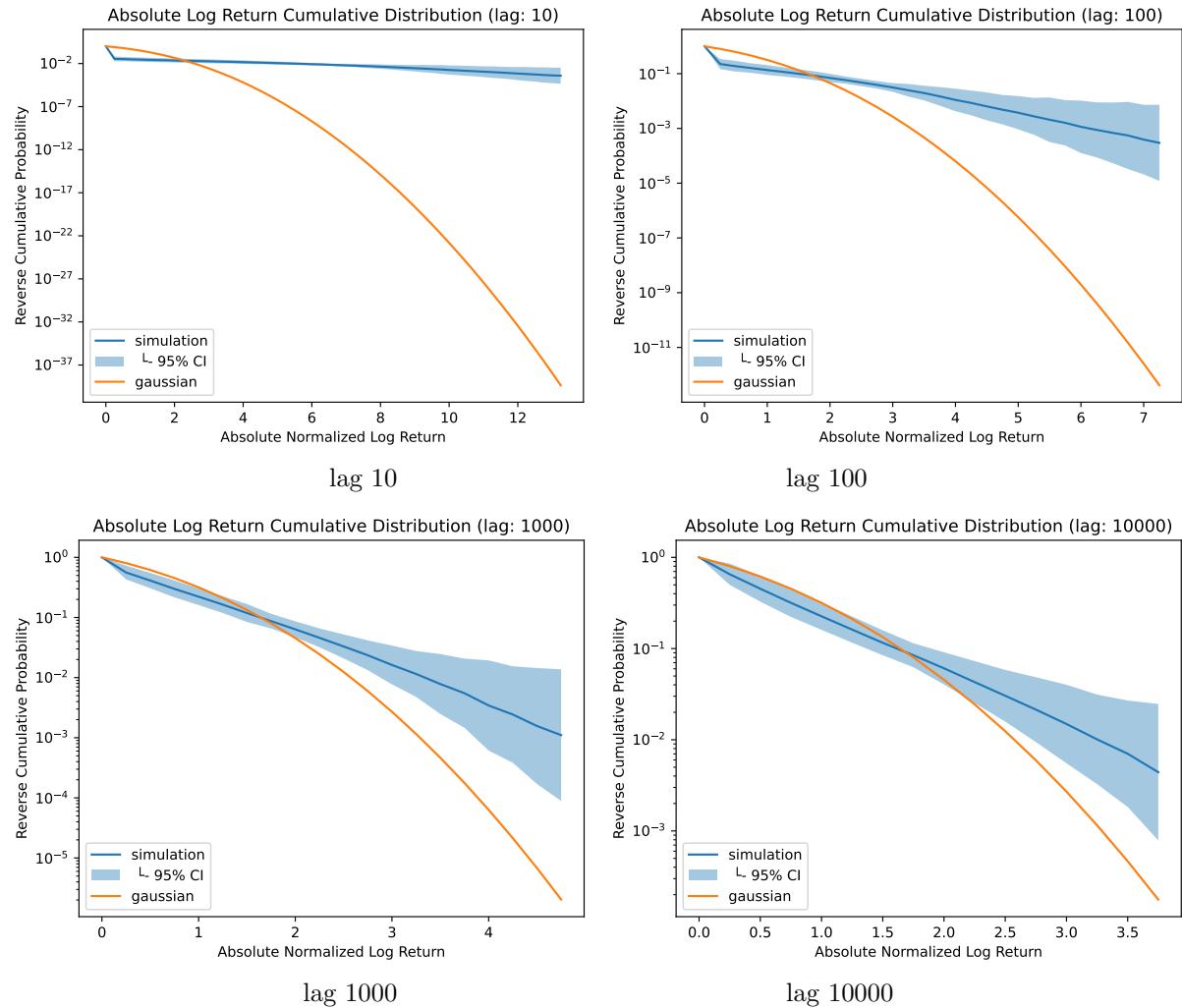


図 4.20: 絶対対数リターンのプロット (Case #4). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は+と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

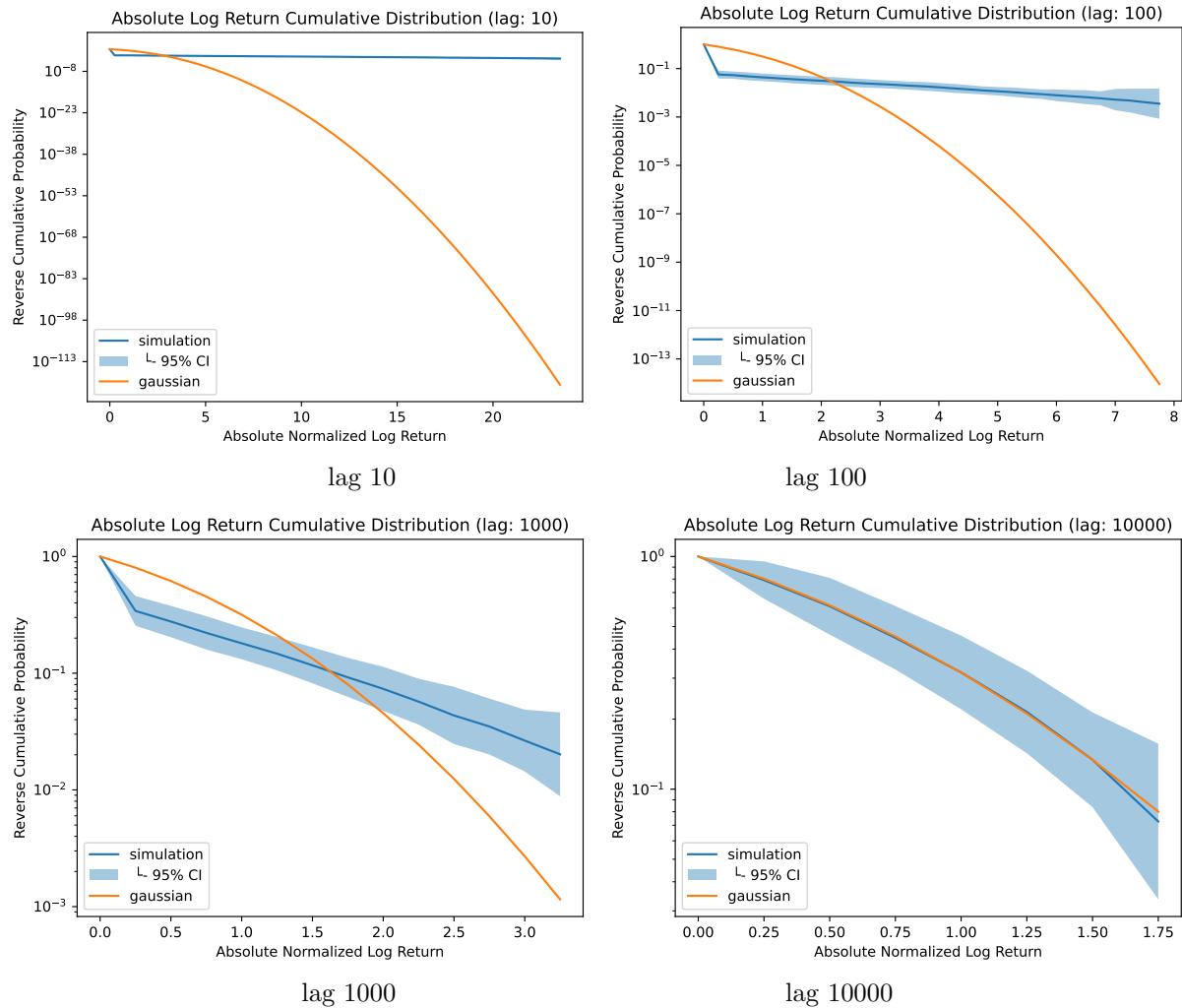


図 4.21: 絶対対数リターンのプロット (Case #5). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+、Aggregational Gaussianity は+と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

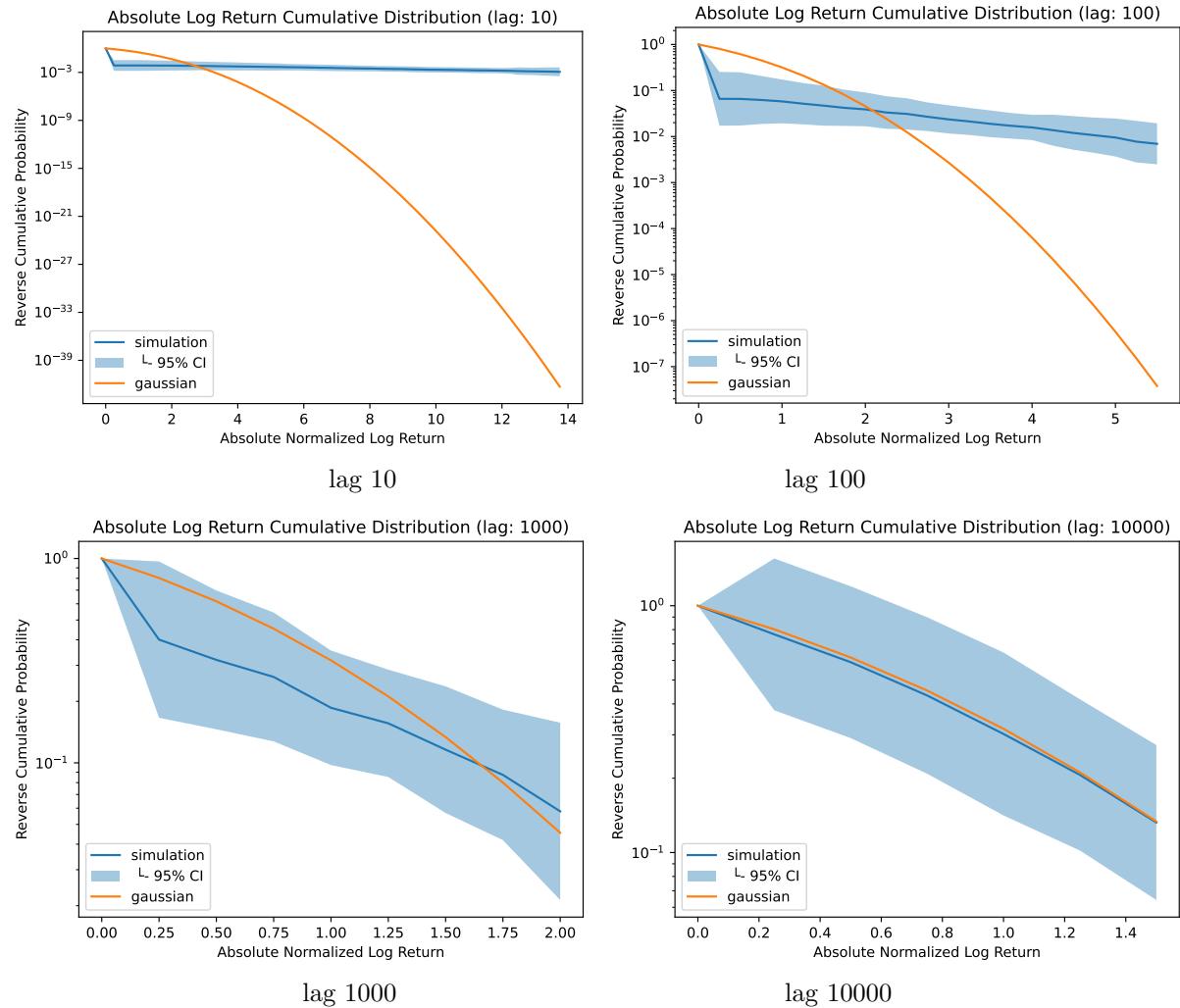


図 4.22: 絶対対数リターンのプロット (Case #6). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている. Heavy Tail Property は+, Aggregational Gaussianity は+と判定. オレンジ色の線はガウス分布の確率密度関数であり, 塗られているエリアは 1000 試行の結果から算出された 95% CI である.

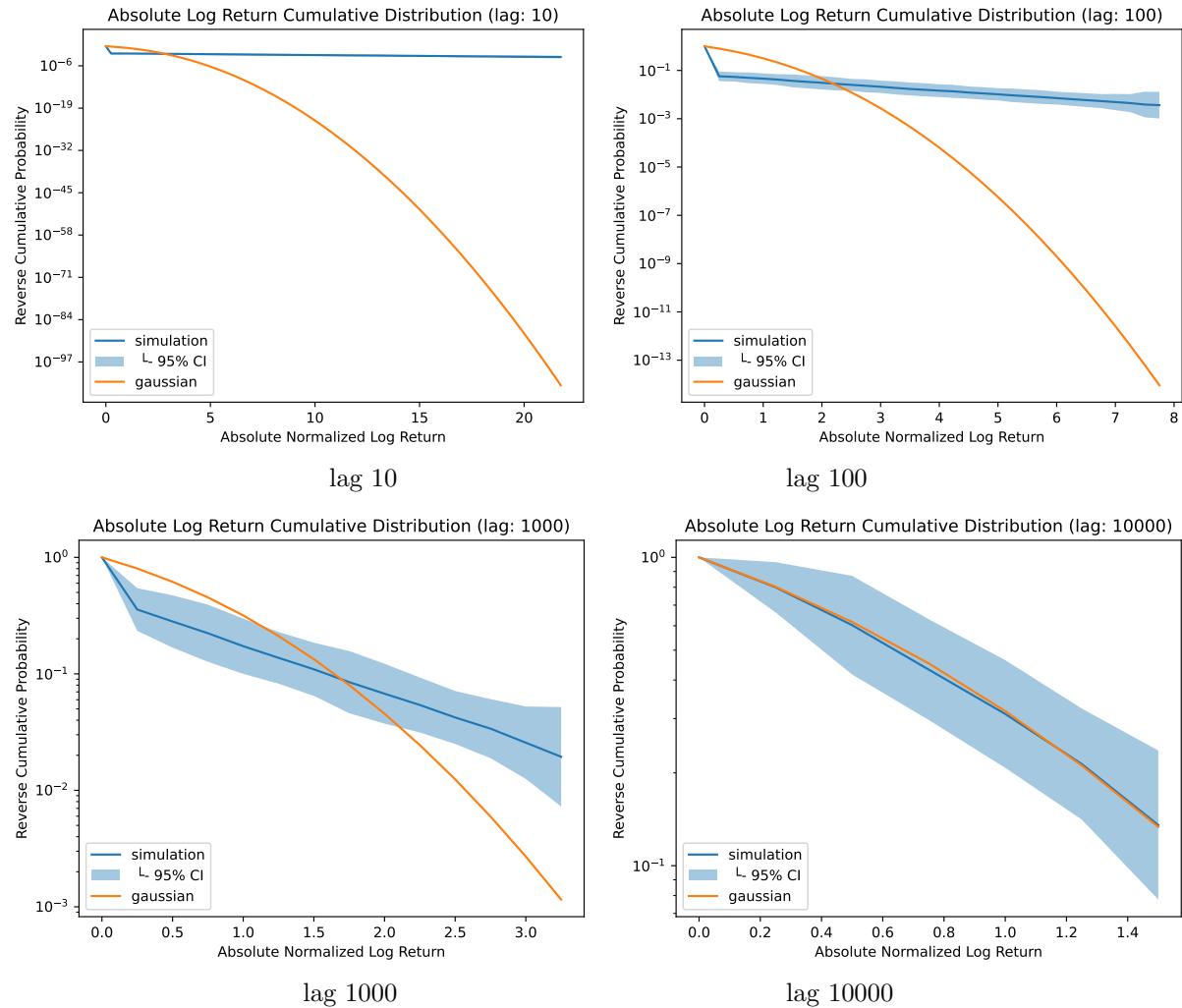


図 4.23: 絶対対数リターンのプロット (Case #7). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は+と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

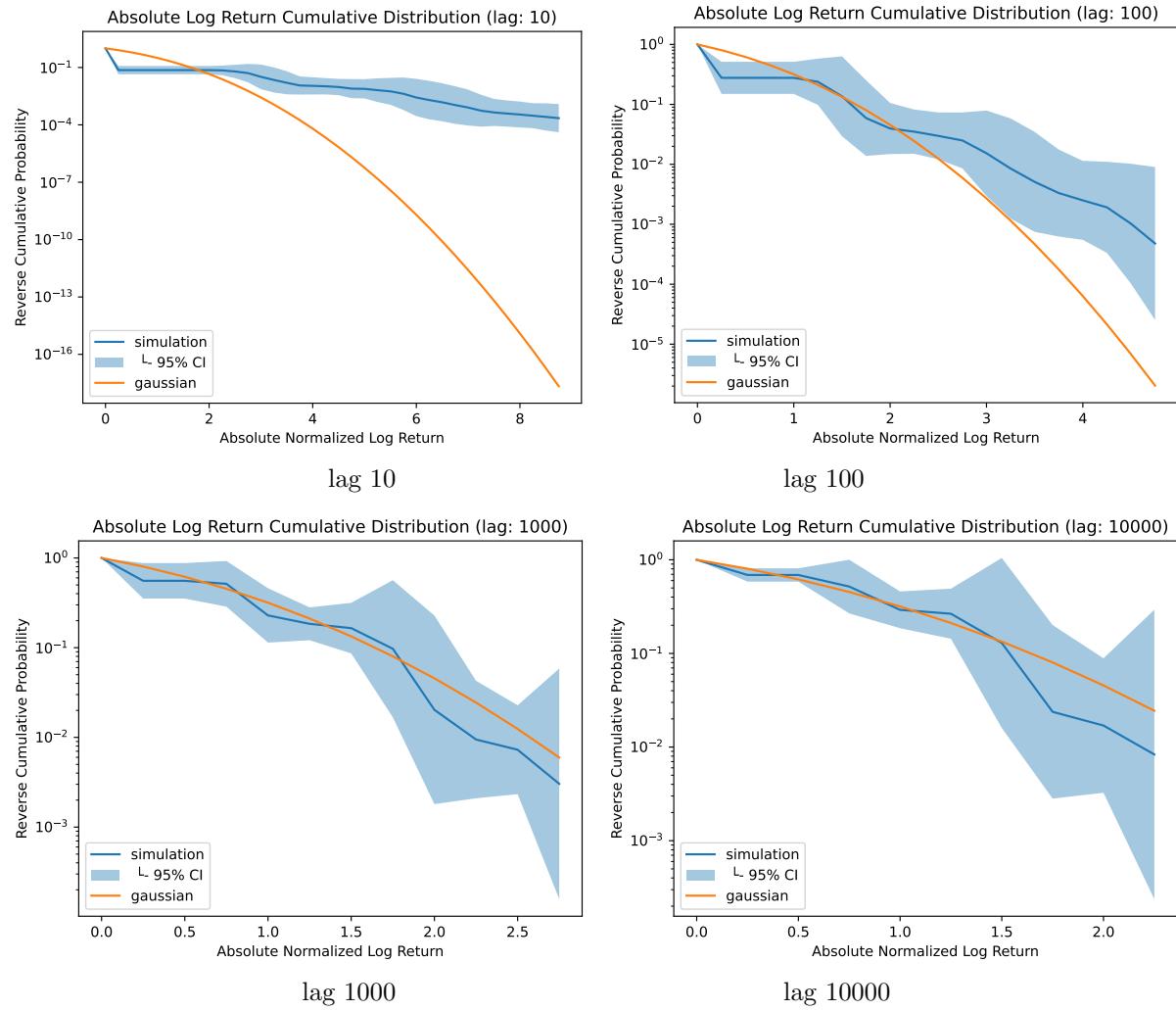


図 4.24: 絶対対数リターンのプロット (Case #9). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は-と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

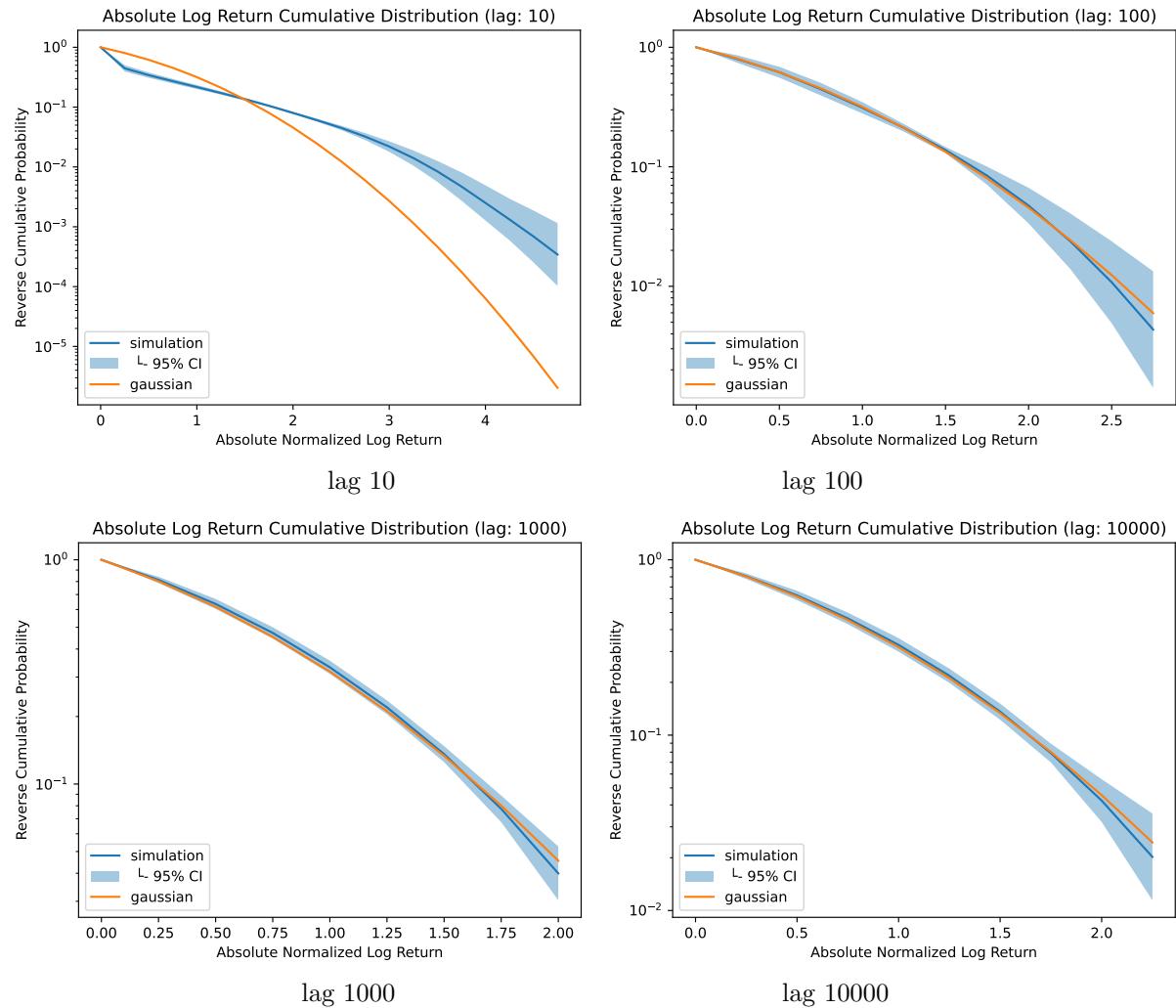


図 4.25: 絶対対数リターンのプロット (Case #10). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は-と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

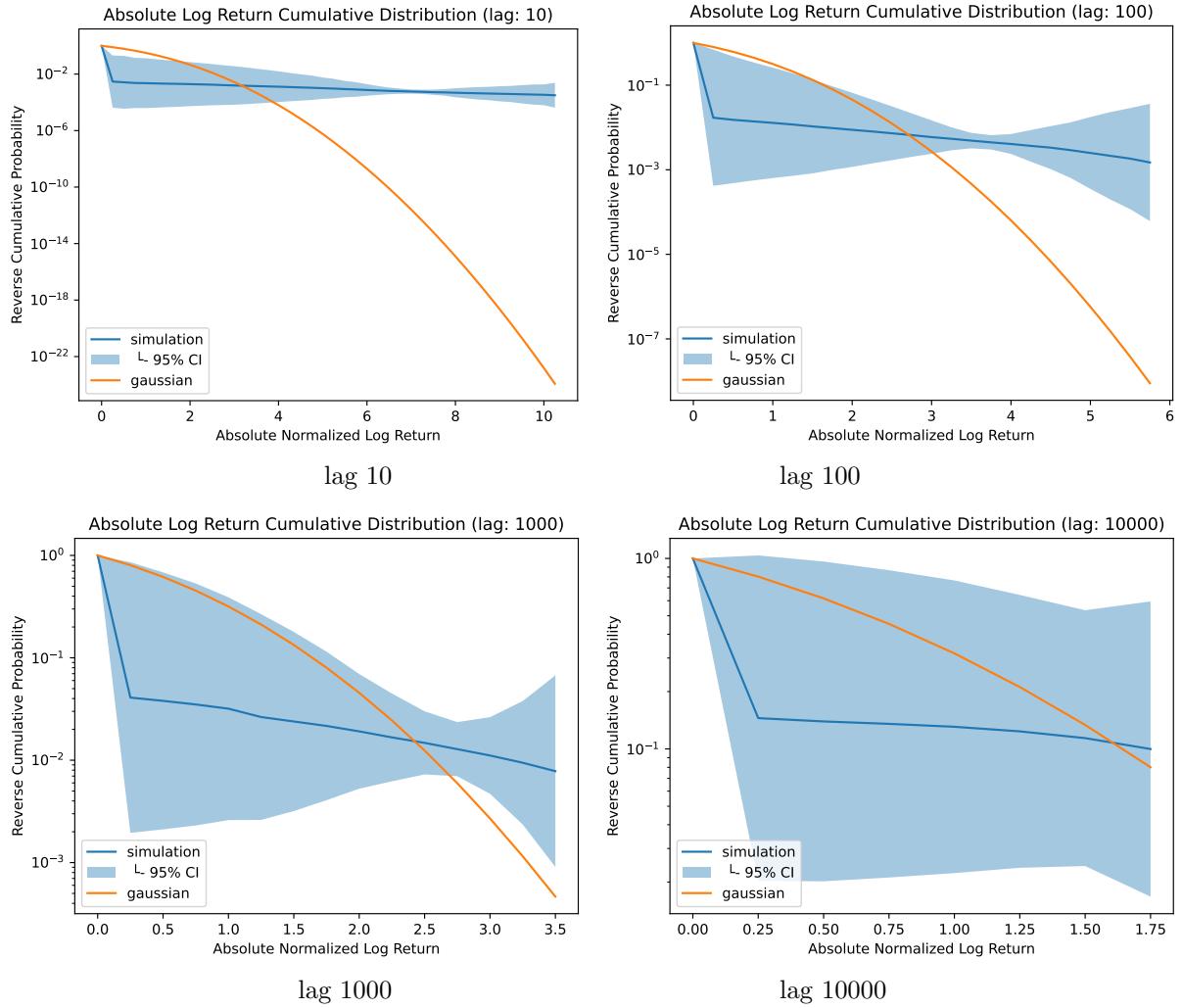


図 4.26: 絶対対数リターンのプロット (Case #11). それぞれは、対数リターンの計算に使用するラグがそれぞれ 10, 100, 1000, 10000 Ticks と異なっている。Heavy Tail Property は+, Aggregational Gaussianity は-と判定。オレンジ色の線はガウス分布の確率密度関数であり、塗られているエリアは 1000 試行の結果から算出された 95% CI である。

Volatility Clustering

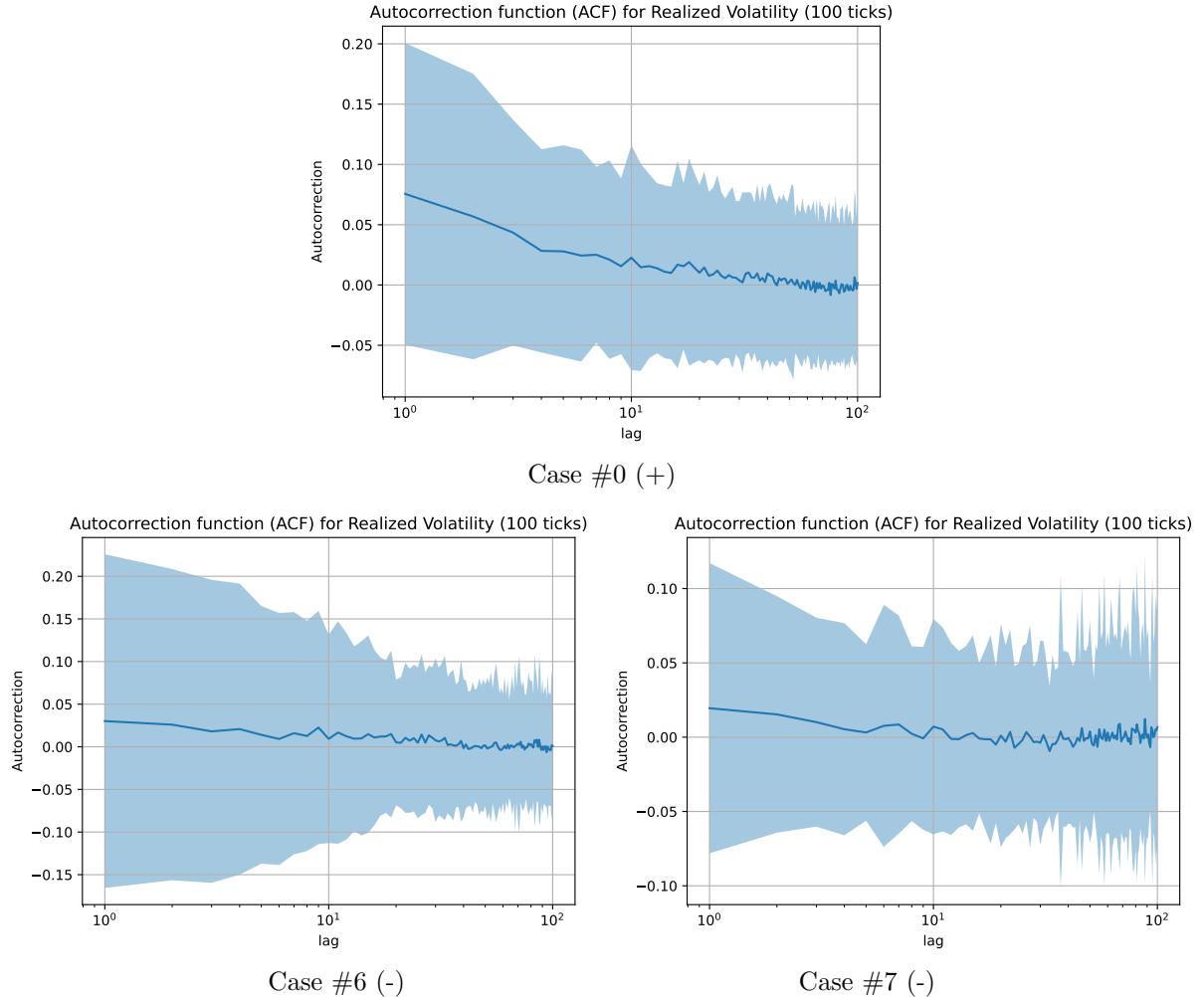


図 4.27: Realized Volatility の Autocorrelation Function (ACF). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。上段のプロットが Case #0 (最適値), 左下と右下がそれぞれ Case #6 と #7 の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

図 4.27 は、Realized Volatility の ACF のプロットである。Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。横軸は、対数リターンの計算に使用するタイムラグであり、このラグに基づいて計算された対数リターン 100 個ずつの二乗平均の二乗根を取ることで Realized Volatility は計算される。先行研究 [82] によれば、Volatility Clustering は、小さめのラグの場合に小さいものの観測ができる。もし、ボラティリティの自己相関が高すぎた場合、オプション価格におけるアービトラージが発生してしまうなど、問題が発生するため、ボラティリティの自己相関は、絶対値としては決して大きくはない。しかしながら、ボラティリティクラスタリングは、確かに存在する現象であるため、

小さいながらも安定的な自己相関が Stylized Facts として期待される。先行研究 [82]によれば、この自己相関の大きさは概ね 0.1 以下の値である。図 4.27 によれば、Case #0 の場合は、これにまさに合致する結果が得られている。一方で、Case #6 と #7 の場合、ボラティリティの自己相関はほとんど観測できない。しかしながら、その判断は非常に曖昧であり、適切なボラティリティの大きさも定量的に分析されているわけでもない。そのため、若干疑念もあることから、Case #6 と #7 に対する判断は、“-”とした。

以下では、他のケースも含めた全ての結果を示す。

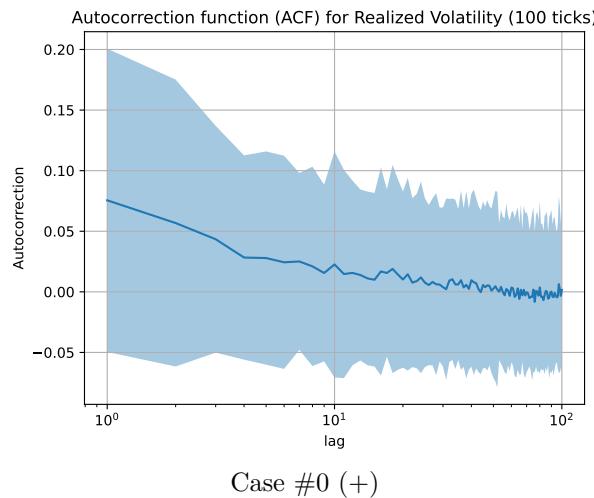
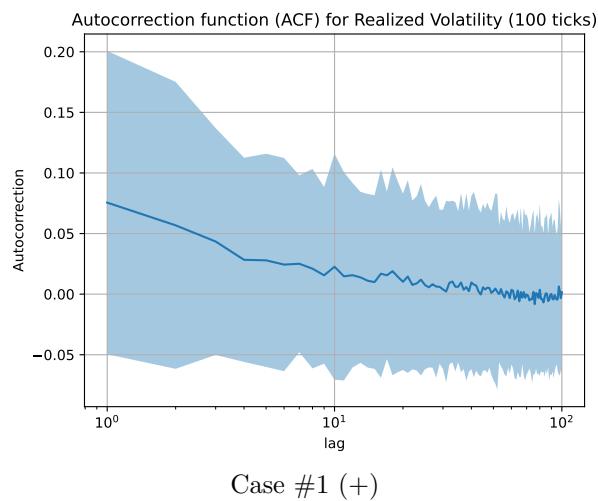
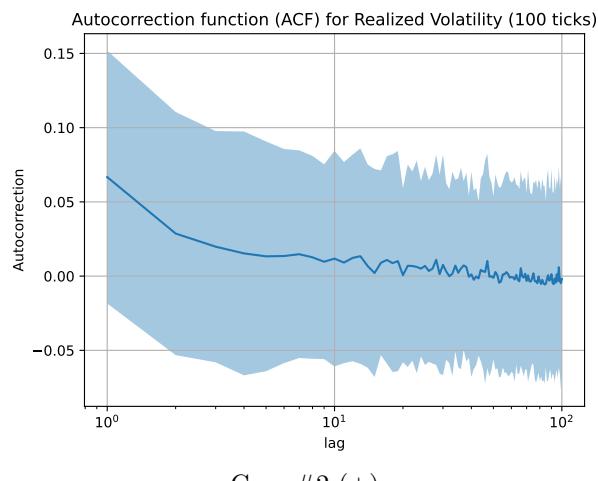


図 4.28: Realized Volatility の ACF (Case #0)。Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。



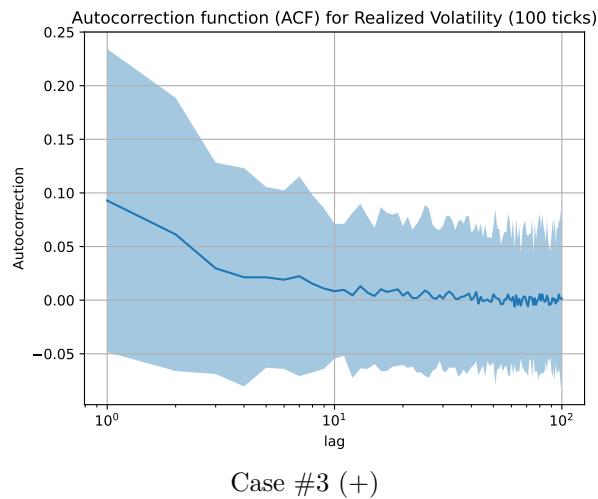
Case #1 (+)

図 4.29: Realized Volatility の ACF (Case #1). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



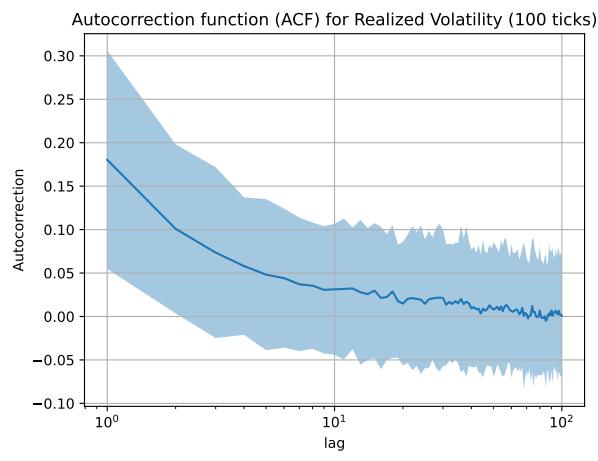
Case #2 (+)

図 4.30: Realized Volatility の ACF (Case #2). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



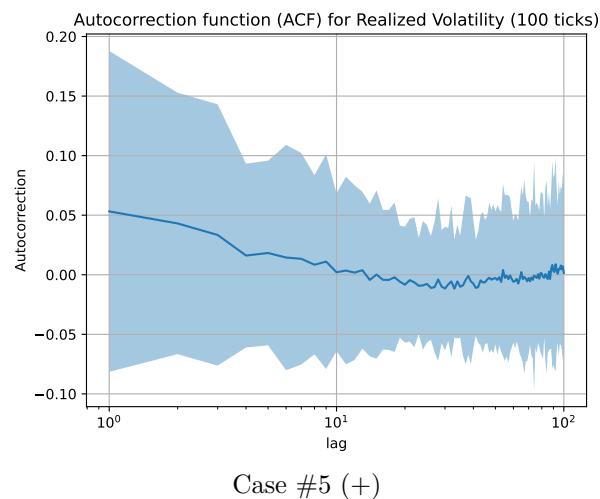
Case #3 (+)

図 4.31: Realized Volatility の ACF (Case #3). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



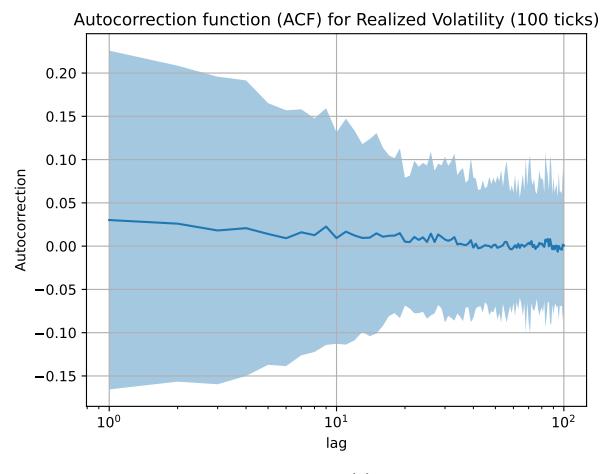
Case #4 (+)

図 4.32: Realized Volatility の ACF (Case #4). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



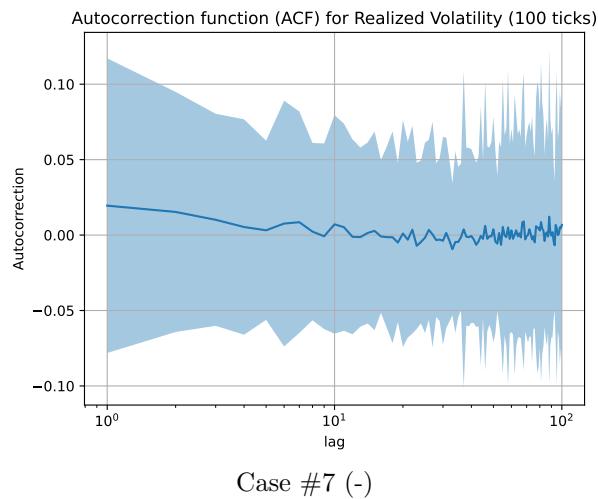
Case #5 (+)

図 4.33: Realized Volatility の ACF (Case #5). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



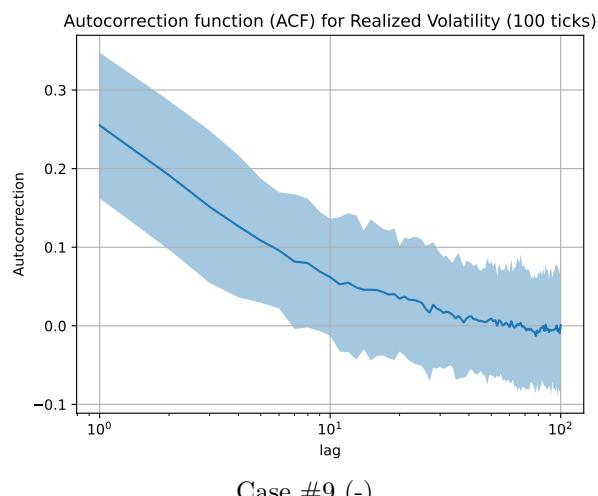
Case #6 (-)

図 4.34: Realized Volatility の ACF (Case #6). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



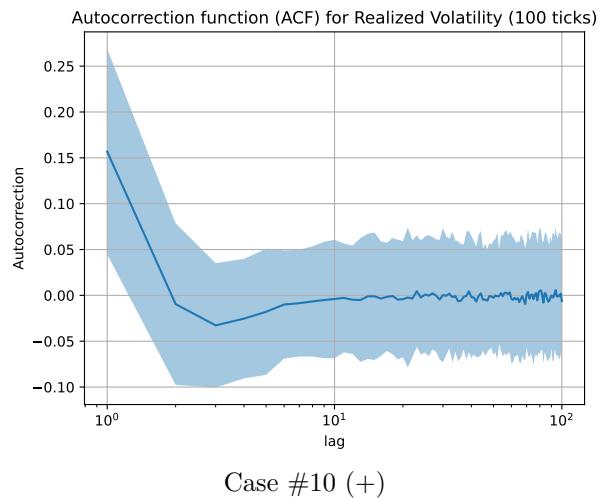
Case #7 (-)

図 4.35: Realized Volatility の ACF (Case #7). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



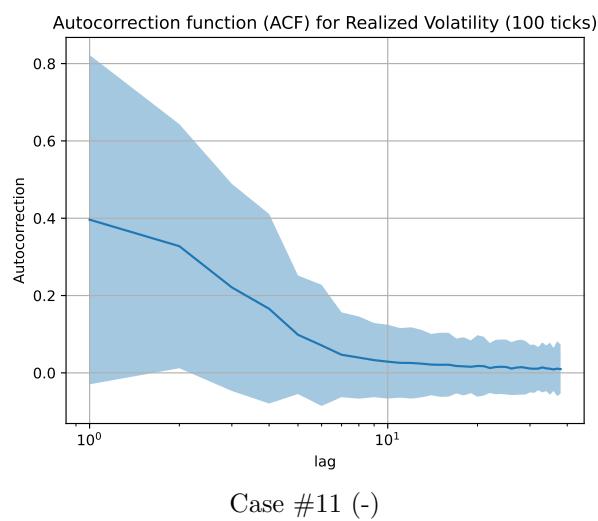
Case #9 (-)

図 4.36: Realized Volatility の ACF (Case #9). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



Case #10 (+)

図 4.37: Realized Volatility の ACF (Case #10). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。



Case #11 (-)

図 4.38: Realized Volatility の ACF (Case #11). Realized Volatility は、100 Tick のタイムウィンドウで計算をおこなっている。塗りつぶされたエリアは1000試行の結果の95% CIである。

Kurtosis/Skewness

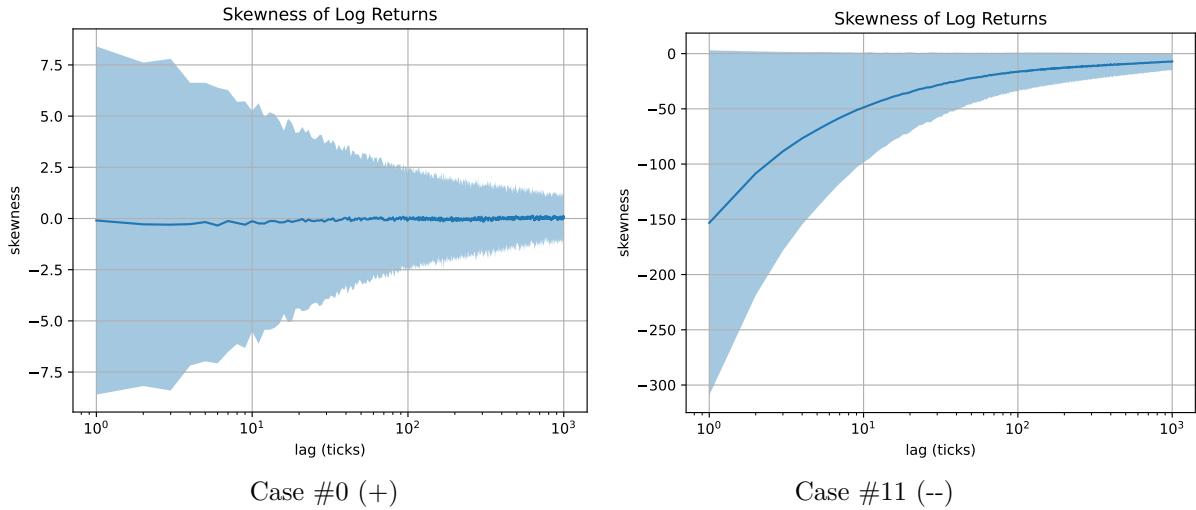


図 4.39: 対数リターンの異なるラグごとの尖度のプロット. 横軸は、対数リターンの計算に使用するラグを示している. 左のプロットは、Case #0 (最適値), 右のプロットは Case #11 の結果である. 塗りつぶされたエリアは 1000 試行の結果の 95% CI である.

前述の通り、尖度 (Kurtosis) と歪度 (Skewness) についても分析を行った. これらは、下記の通り計算される.

$$x_t := \ln \{p_t/p_{t-1}\}, \quad (4.10)$$

$$\text{skewness} = \frac{1}{T} \sum_{t=1}^T \frac{(x_t - \bar{x})^3}{(x_t - \bar{x})^{2\frac{3}{2}}}, \quad (4.11)$$

$$\text{kurtosis} = \frac{1}{T} \sum_{t=1}^T \frac{(x_t - \bar{x})^4}{(x_t - \bar{x})^2}, \quad (4.12)$$

図 4.39 は Skewness についての結果のグラフの一部である. 本研究で用いているシミュレーションでは、成長率を考慮に入れない対称なモデルであるため、歪度は 0 になるのが自然である. 図 4.39においては、Case #0 が、この条件を満たしている一方で、Case #11 は、非対称な結果となっており、条件を満たしておらず、--として判断した.

以下では、残りのグラフも掲載する、

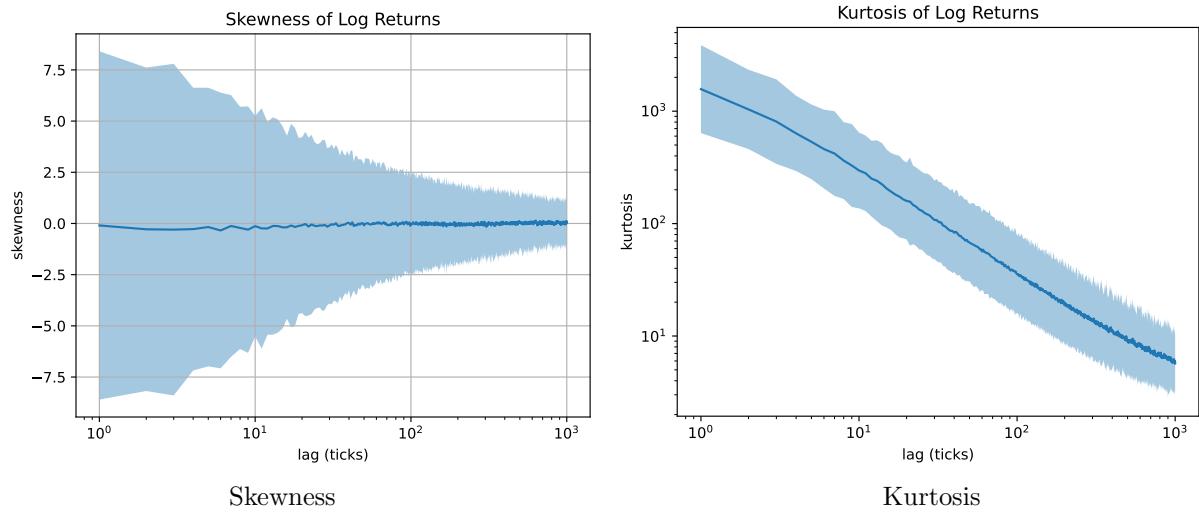


図 4.40: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #0, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

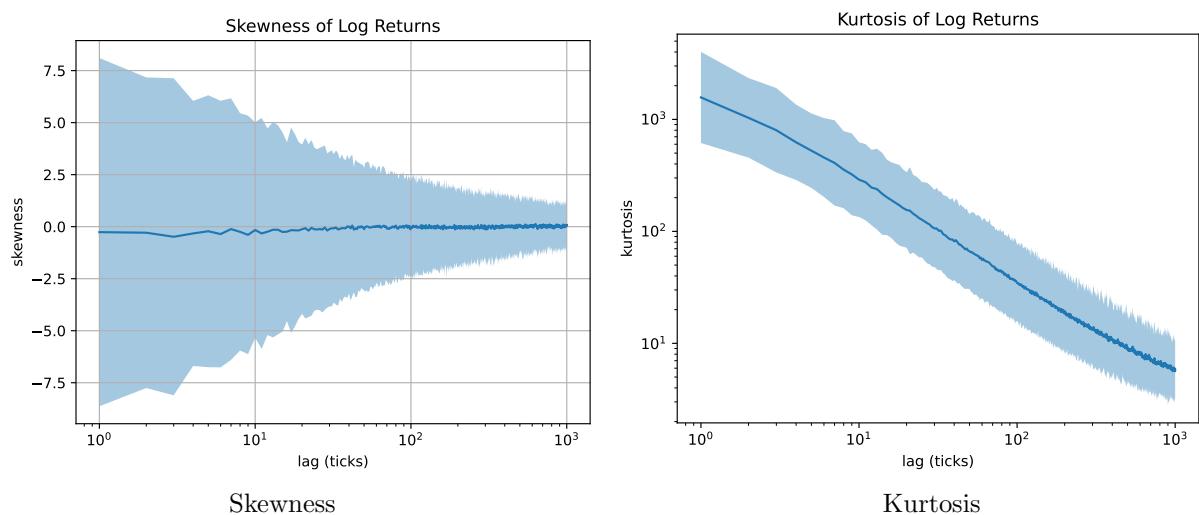


図 4.41: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #1, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

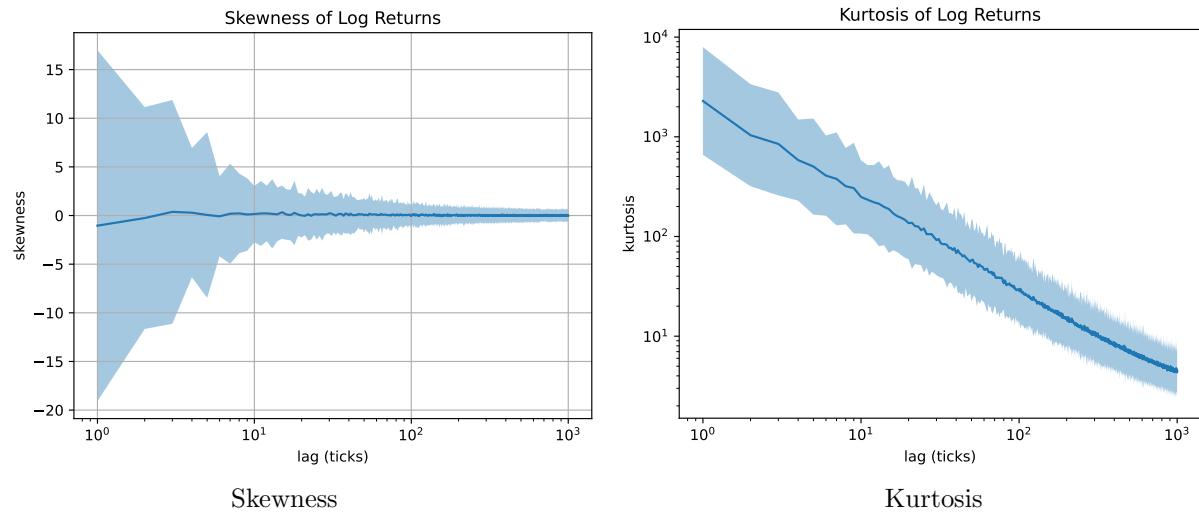


図 4.42: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #2, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

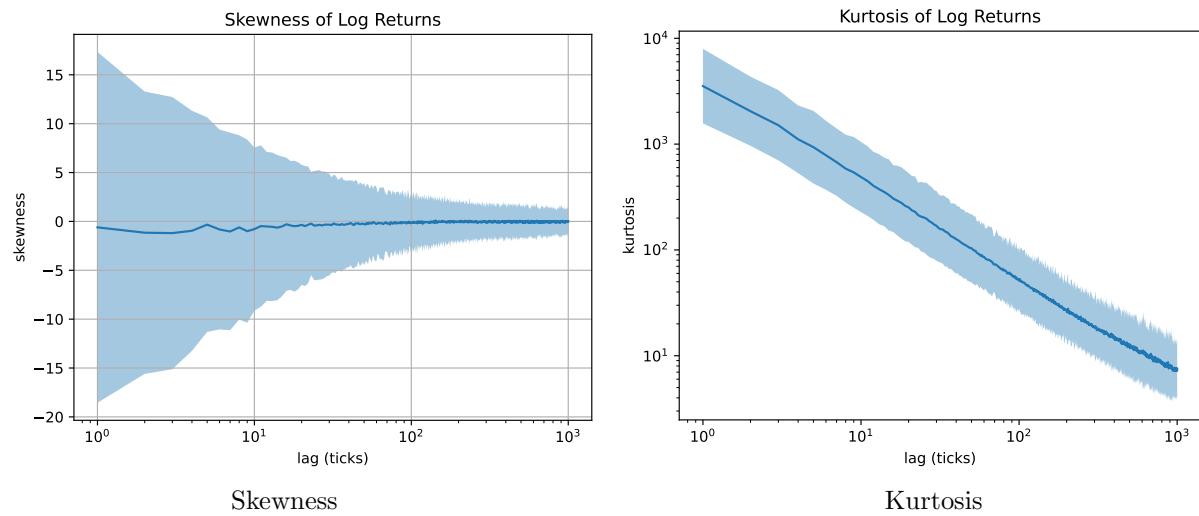


図 4.43: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #3, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

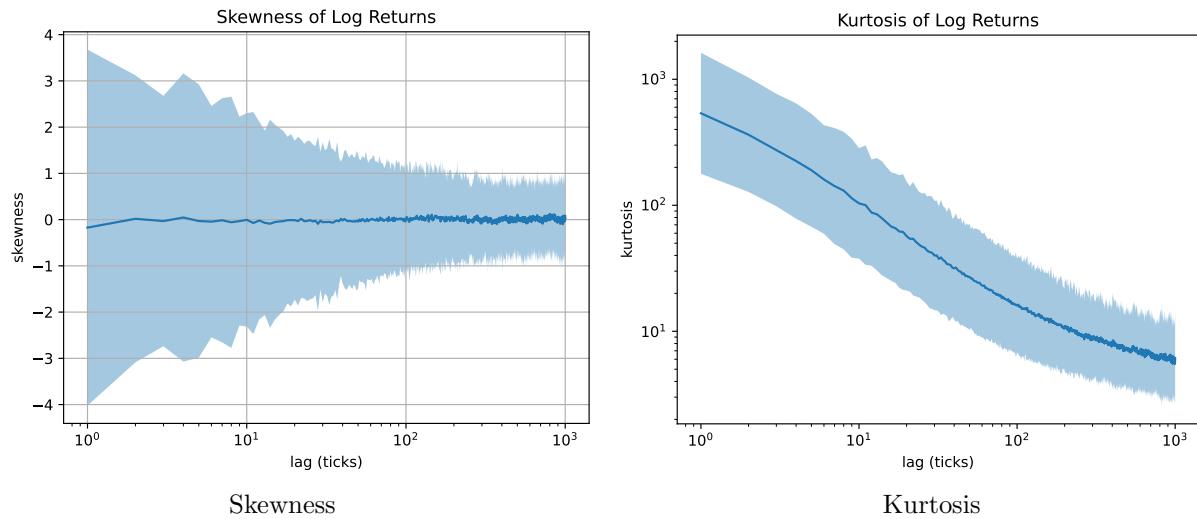


図 4.44: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #4, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

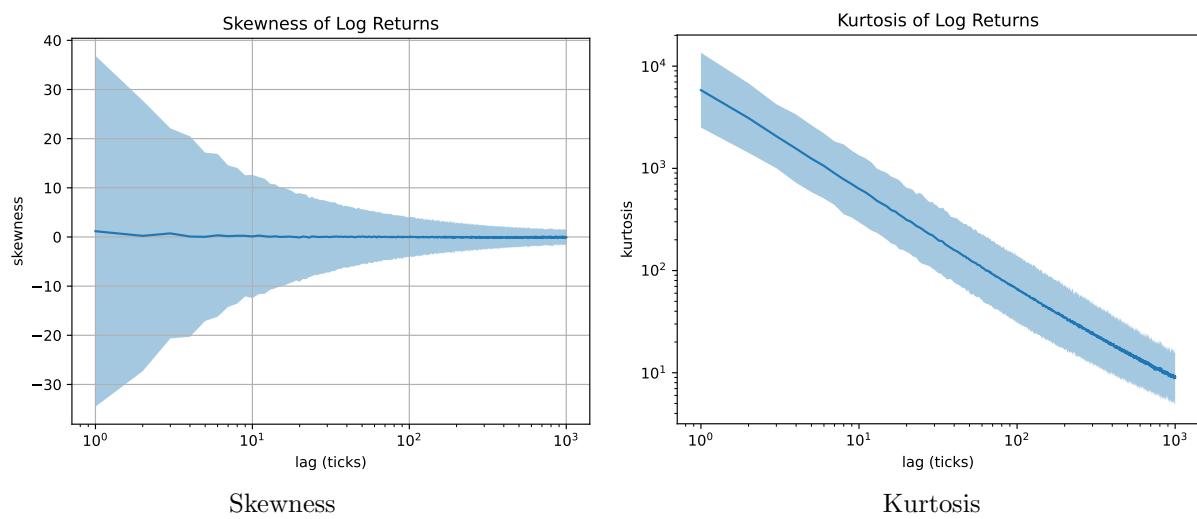


図 4.45: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #5, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

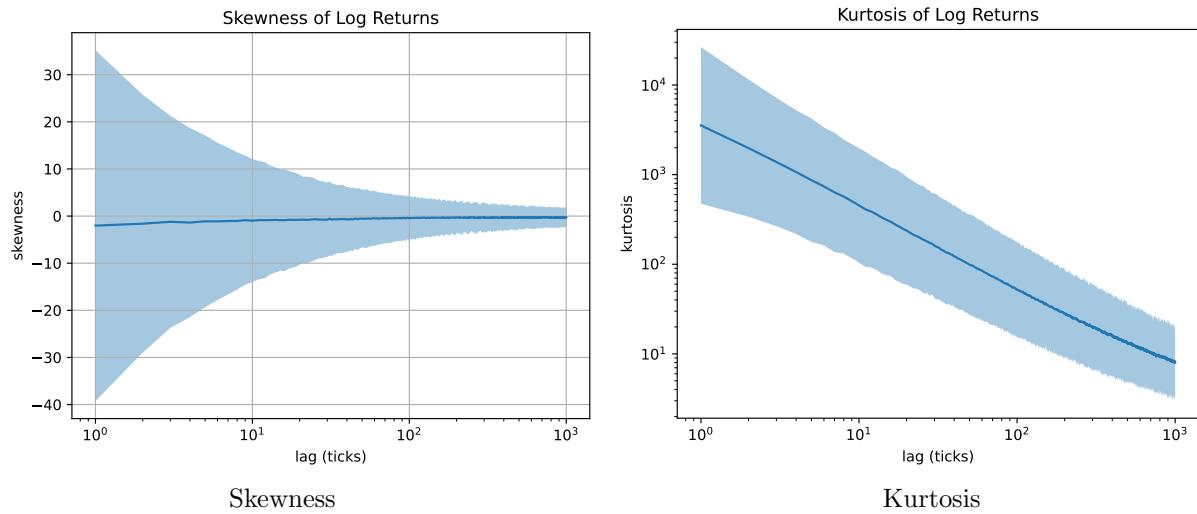


図 4.46: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #6, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

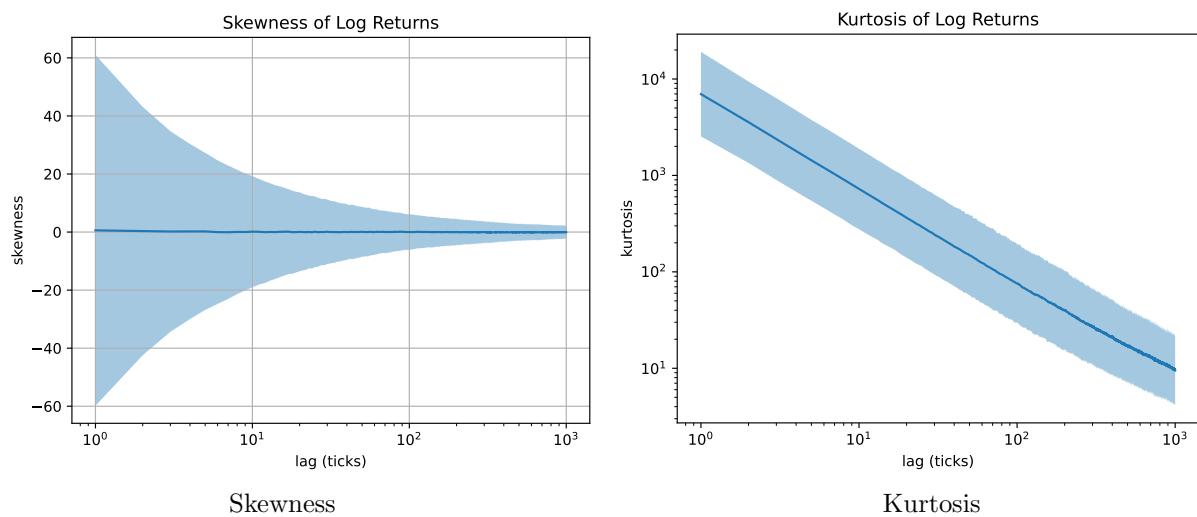


図 4.47: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #7, +). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

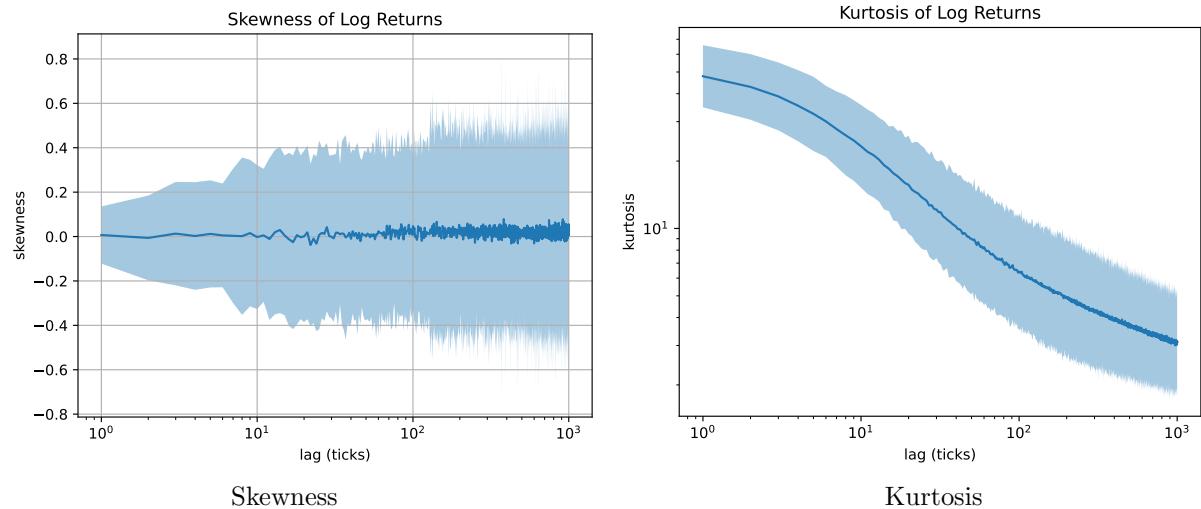


図 4.48: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #9, -). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。尖度が小さすぎるため-と判定。

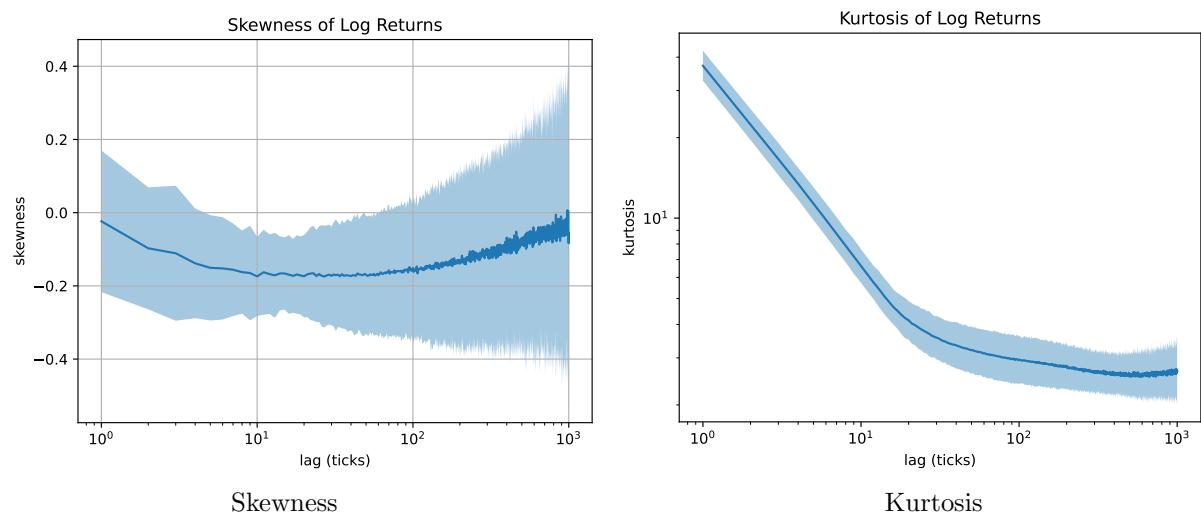


図 4.49: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #10, -). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

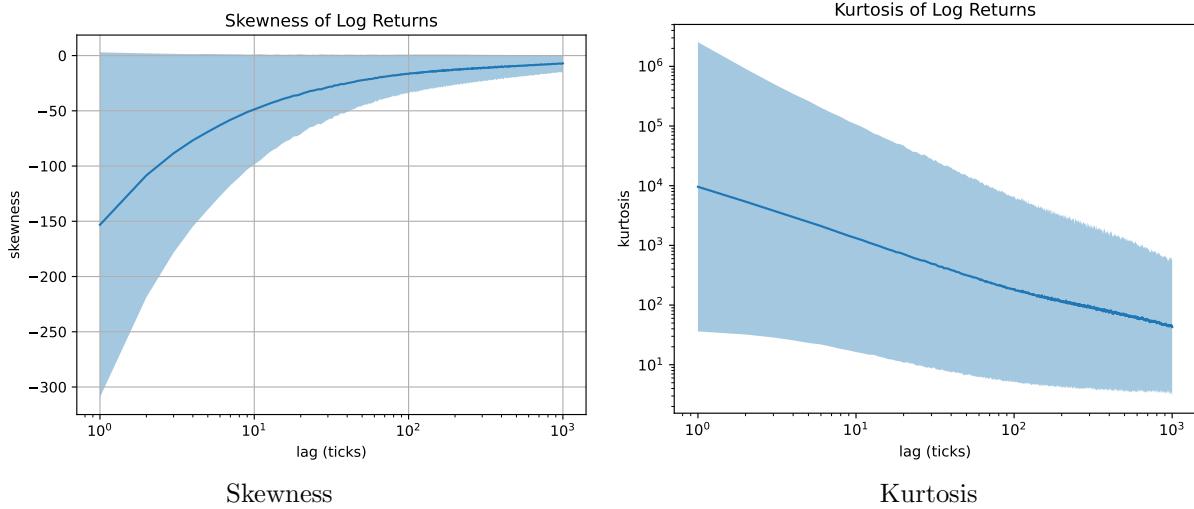


図 4.50: 対数リターンの異なるラグごとの尖度と歪度のプロット (Case #11, -). 横軸は、対数リターンの計算に使用するラグを示している。左のプロットは歪度の結果、右のプロットは尖度の結果である。塗りつぶされたエリアは 1000 試行の結果の 95% CI である。

4.6 考察

第一に、結果に基づけば、GAN Evaluator は、従来の伝統的な Stylized Facts に基づく定性的評価の代用として有効であると言えると考える。直感的理として、GAN は、Critic を通じて、データに基づいたあいまいな分布を獲得することに成功しており、それを活用することによって、良い評価指標として機能していると理解できると考えられる。図 4.1 によれば、GAN evaluator の出力は、従来の定性評価と相關を持ったものとなっており、また、出力が Stylized Facts の再現の有無という段階的なものから、-1 から 1 の間の連続値での評価に置き換えることができている。そのため、曖昧性もはらむ定性評価と比較して、より使い勝手の良い定量的指標を作成できていると言えるのではないか。

また、GAN Evaluator を用いたパラメータチューニングも適切にチューニングができたことは、GAN Evaluator の出力に一定の妥当性があることを示していると考えられる。アプリケーションテストの結果に基づけば、Stylized Facts の再現に非常に重要ないくつかのパラメータがあることが分かった。たとえば、ファンダメンタルファクターの重みだけを変えた場合 (Case #11) でも、Stylized Facts の棄損が確認できた。つまり、たった1つのパラメータを変えただけでも、Stylized Facts が再現できないということである。これは、最適パラメータの範囲が広くではなく、チューニングの結果はそれなりに妥当である簡単ではないものであると言える。この結果に基づけば、GAN Evaluator の出力は、現実的なシミュレーションの追求においてはよい指標であるということが分かった。

一方で、GAN Evaluator の限界も明らかになっている。Case #8においては、GAN Evaluator の出力は 0.64020 であったが、シミュレーション自体は全くおかしなものであった。Case #8においてはエージェント数を 2 に変更したことにより、注文の流動性が枯渇

し、価格の変動が起きなくなったことが原因であった。しかしながら、実際の市場においても、流動性が低くなる時間帯は存在する。そのため、実市場の流動性の低い時間と、Case #8のシチュエーションは類似しており、判断が難しいのではないだろうか。問題は、その流動性の低い時間が全時間にわたるかどうかであるという点であり、その点については、現時点のGANでは解決できていないというわけであると考えられる。PGSGANにおいては、過去のたった20注文のみを入力として取っているため、この判断が容易ではない。この点に関しては、GANを変更することで解決できる可能性もあると言えるので、必ずしもGAN Evaluator固有の問題ではない可能性もある。

続いて、評価分析について考察する。本研究での分析では、評価において、Stylized Factsという曖昧なものを使用しているために、バイアスや主観を含んでしまう可能性は十分にある。この問題に対しては、もっと客観的な評価基準を用いて、より精密なGAN Evaluatorの評価を行いたいところではある。しかしながら、定量的評価が難しいために、GAN Evaluatorを用いて、より正確な評価を行うことが本研究の目的であり、トートロジーを引き起こす。つまり、定量的評価が難しいから、本研究では代替的な定量的評価手法を提案しており、その評価をより客観的に行うためには定量的評価手法が必要であるというトートロジーが発生する。このトートロジーを解消するためには、完全にコントロール可能なトイプロブレムを使うなどの、他の評価方法を採用しなければならない。一方で、トイプロブレムを使用すると、現実性という観点での評価が困難になるため、難しいところである。

本研究を通じて、GAN Evaluatorの有効性については一定確認できたが、今後の課題も明らかになった。まず、様々なシミュレーションでの実験が必要であろう。本研究では、人工市場シミュレーションを用いて提案手法を評価した。しかしながら、提案手法は、様々なシミュレーションに適用可能であると考えており、他の各種社会シミュレーションでもテストし、その有効性を検証することが今後の課題である。また、本研究では、シミュレーションモデルは固定し、パラメータのみをチューニングの対象とし、Optunaを用いたパラメータチューニングを行なった。Optunaは深層学習のパラメータチューニングに頻繁に使用される技術であるが、近年、このパラメータチューニングに加えて、深層学習のアーキテクチャ自体をチューニング使用という、AutoMLやNeural Architecture Search (NAS) [143]などの技術が提案されている。このNASと本研究の提案手法のアナロジーを考えると、シミュレーションモデルのアーキテクチャ自体をGAN Evaluatorの評価に基づいてチューニングしてしまうということも考えられる。

4.7 本章の結論

本研究においては、社会シミュレーションの評価における従来の定性評価を置き換える、GANを用いた定量的評価手法を提案した。社会シミュレーションの評価においては、典型的な現象が再現できているかという観点で評価を行うのが一般的であり、曖昧な現象を評価するために、定量的評価は一般的に困難である。しかしながら、定量評価が不可能で

あると、モデルパラメータのチューニングや、シミュレーション間の直接比較などが困難であるという問題がある。そこで、GANを活用し、GANが実データの曖昧な分布特性と学ぶことができると仮定し、GANを定量評価に活用する手法を提案した。実験では、人工市場シミュレーションを用いて、GANによる定量評価の性能評価を行った。その結果、GANによる定量評価を用いて、シミュレーションのパラメータチューニングが可能であることを示し、アブレーションテストを通じて、その妥当性を確認した。加えて、従来の定性評価との比較を行ったところ、GANを用いた定量評価は、従来の定性的評価との十分な一致が確認できた。このGANによる社会シミュレーションの定量的評価指標は、他の社会シミュレーションにも適用可能であると考えられ、今後の課題として検証されたい。

第III部

シミュレーションを用いたデータマイニング
手法の評価

第5章 人工市場データマイニングプラットフォーム

5.1 金融データマイニングにおける問題点と本章の研究背景

金融市場においては、多くのトレーダーが様々な方法で、利益を上げようと取り組んでいる。近年、この目的で、機械学習などが使用されることが多くなった。学術研究においても、様々な手法が提案されることが増えている。

一方で、深層学習も非常に良く使用されるようになってきており、その手法も洗練されてきている。例えば、Wang ら [3] は、金融市場のトレンド予測に対して、Convolutional Long Short-term Memory (LSTM)-based Variational Sequence-to-sequence Model with Attention (CLVSA) というモデルを提案した。このモデルは非常に多くの深層学習機構を取り入れたものとなっている。

さらに、近年の深層学習の普及は、金融市場において、利益を上げることをより難しくしているとも考えられる。より高度な技術の普及は、簡単な技術による利益の獲得可能性を駆逐し、予測の難易度を高め、利益を得にくくすると考えられる。これは、議論の余地はあるものの、効率的市場仮説 [144] の文脈でも理解できる。

近年、非常に多くの予測モデルが提案される中で、それらの評価は決して充分に適切に行われているとは言えない。特に、予測性能の評価に関しては、既存のモデルよりも良いかどうか、という点にのみフォーカスが当てられており、その評価自体もバックテストに基づいたものである。確かに、バックテストはモデルの比較に有効であるものの、常に過去のデータに依存している点で、将来の性能を保証するものではなく、必ずしも完全とはいえない。また、将来的により高性能な予測モデルの普及に伴って、提案手法の予測能力が大幅に低下する可能性もある。そのため、バックテストに基づいた評価手法というのは疑わしい点も存在する。加えて、それぞれの研究が個別に行われているため、評価軸も一貫していないという問題点も存在する。

そのため、金融市場をどの程度予測できるのかを理論的に分析したり、どの程度理想的な状況において予測が可能であるのかということを明らかにすることは、とても重要であると考える。もし、評価実験が完全にコントロールされた状況で行われれば、同じ条件での評価ということが実現可能になる。例えば、よくあるケースとして、古いデータを用いた評価は、近年の高度な予測モデルによって既に利益を獲得できるチャンスが潰されていることが少ないために、高い性能を出しがちである。そのため、完全にコントロールされ

ている環境であれば、この難易度もコントロールされているために、直接的な比較が可能になる。

人工市場シミュレーションによるアプローチはこの問題を解決できると考えられる。そこで、本研究では、データマイニングをコントロールされた環境である人工市場内で行う、「人工市場データマイニングプラットフォーム」を提案する。人工市場シミュレーションは、実験ごとに見たいものに合わせてシミュレーションを組むことが可能である。そのため、データマイニング手法を特定の環境下でテストすることが可能である。加えて、人工市場モデルを変更することで、様々なシナリオをテストすることができる。例えば、ティックタイムレベルのシチュエーションのテストも、外部から何かしらの影響を受けた場合のシナリオのテストも行うことができる。

本章においては、人工市場データマイニングプラットフォームのコンセプトの提案と、その有効性の確認のための実験を行う。ここでは、ティックスケールのシミュレーションを用いて、基本的なデータマイニングのモデルの性能の分析を行うことで、提案コンセプトの妥当性を確認する。使用するデータマイニングモデルは基本的なもので、その結果には特に意外性のある結果を得られるものではないが、この分析を通じて、コンセプトの妥当性と、プラットフォームを使用した場合の重要なファクター分析の実現可能性について示す。

5.2 人工市場データマイニングプラットフォーム

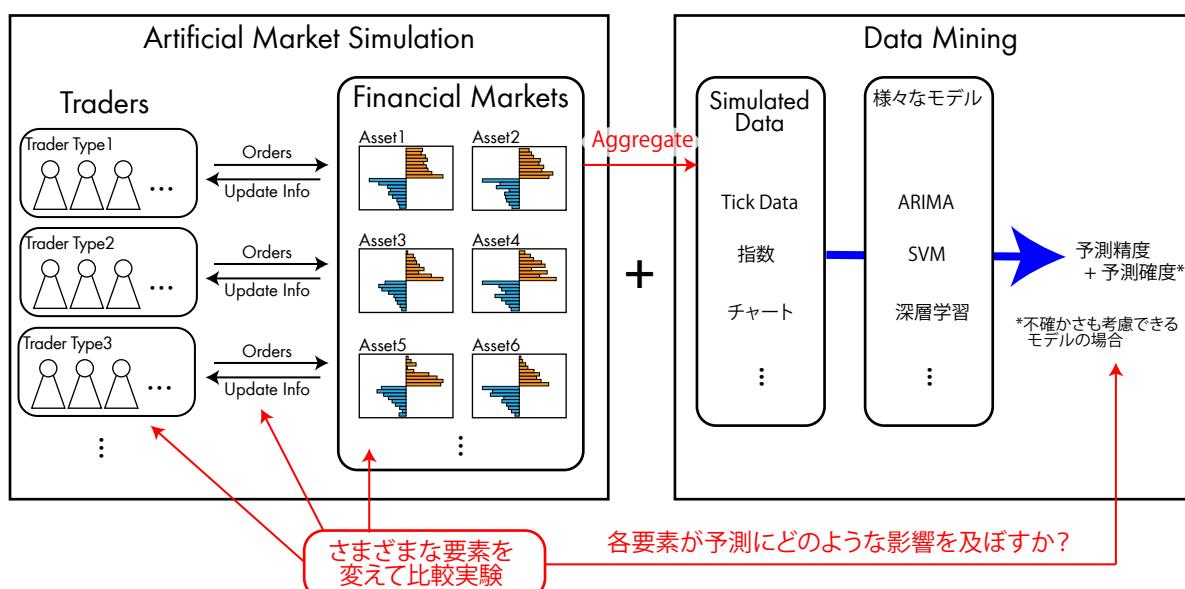


図 5.1: 人工市場データマイニングプラットフォームの概要

人工市場データマイニングプラットフォームは、人工市場シミュレーションとデータマ

イニング手法を組み合わせたものであり、あくまでコンセプトでしかないので、その詳細については、それぞれ変更可能である。図5.1にその概要を示す。

人工市場シミュレーションは、モデルの構築がとても重要である一方で、このプラットフォームは概ねいかなるモデルもデータマイニング手法も適用可能であるため、テストしたい要素に合わせてシミュレーションモデルとデータマイニング手法を変更することで、目的の調査を行うことが可能である。例えば、ティックタイムレベルのデータマイニング手法を試したい場合、人工市場シミュレーションもティックタイムレベルのシミュレーションとして実装すれば、ティックデータのマイニングを実現可能である。他にも、もし、外部イベントの影響を分析したいのであれば、外部の影響を再現することのできるシミュレーションを構築し、外部情報を考慮に入れるトレーダー向けデータマイニング手法を使用すれば、目的を達成することができる。

このプラットフォームのメリットは、人工市場を通じて、環境を全てコントロール可能でかつ、観測可能であるという点である。そのため、人工市場の特定の要素のみを変更したシナリオ分析を行うことで比較分析を行うことが可能である。一般に、データマイニングの手法は、実際の過去データでテストされることが多い。そのため、もし、データ期間が短かった場合に、様々な局面をテストすることもできない。特に、金融市場の場合、定常性がないために、多くの局面を学習に用いることはとても重要である一方で、一部の過去データだけではデータが不足している場合もある。そのため、人工市場のパラメータを変えて、様々な局面でデータマイニング手法をテストすることはとても価値がある。これは、バックテストでも確認できることである。

このように、提案手法は、理想的にコントロールされた環境下でデータマイニング手法を検証することが可能で、様々な応用が可能であるプラットフォームになっていると考えている。また、このプラットフォームを用いることで、金融市場のどの要素がトレンドや価格変動の予測を難しくしているのかということを推定することにも利用可能であると考えている。現状では、予測が難しいことは、効率的市場仮説[144]により説明されるとされている。しかしながら、金融市場の予測の難しさは、市場の効率性だけからくるのではなく、様々な要素から発生していると理解するべきではないだろうか。そこで、このプラットフォームを用いることでこの問題への理解を深めることができるのでないかと考えている。

この節では、主にコンセプトについて説明したが、次の節では、実際のモデルの一例を示すとともに、実験についても示す。

5.3 人工市場データマイニングプラットフォームによる実践

この節では、実践的なモデルの一例を示す。ここでは、ティックタイムレベルのシミュレーションとデータマイニングを用いる。

5.3.1 人工市場シミュレーションモデル

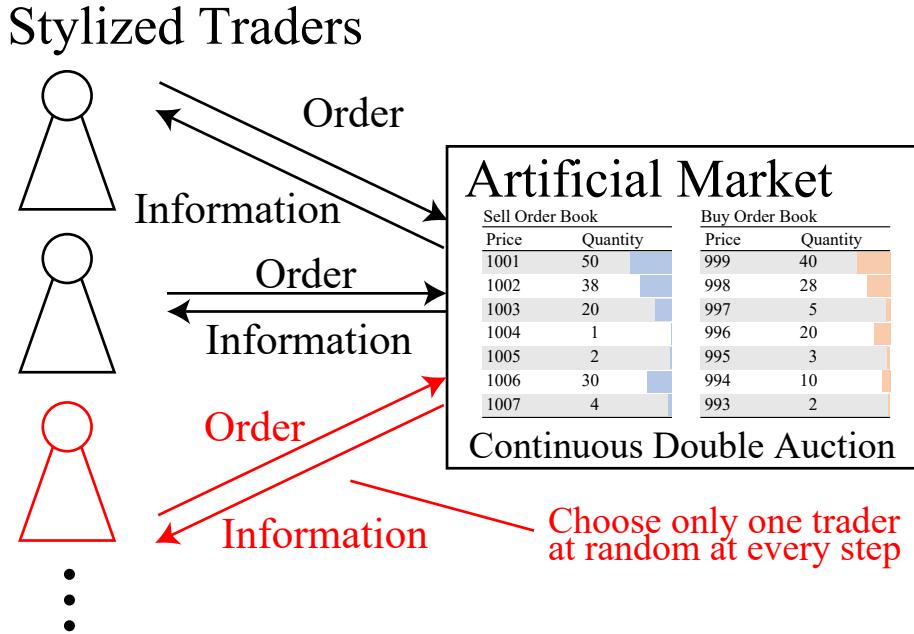


図 5.2: 人工市場モデルの概要. 複数の Stylized Traders と, 1つの人工市場が存在し, 各ステップで, 1つの Stylized Trader が注文機会を与えられ, 注文を出す.

本章で使用する人工市場シミュレーションの概要是, 図 5.2 に示されている通りであり, 1つの連続ダブルオークション市場といくつかの Stylized Traders からなるシミュレーションである. 以下で, その詳細を説明する.

また, シミュレーションモデルの構築にあたっては, “PlhamJ” [58] を使用した.

市場モデル

このシミュレーションでは, 単純化のために, 連続ダブルオークションに基づく市場が 1つのみあると仮定した. 市場の開始時のファンダメンタル価格を 500 に設定した. 市場には, N_{agents} 体の Stylized Trader エージェントが存在し, 各ステップで, 1つのエージェントのみがランダムに選べ, 注文行動を行うことができる. このモデルが, 実際の注文タイミングのランダム性を再現しており, “PlhamJ” [58] の実装と鳥居ら [53] の研究に基づいている.

市場は, 100 ステップのプレオープニング・セッションと, 100,500 ステップのザラ場のセッションからなる. プレオープニング・セッションは, 注文のみを受け付ける期間であり, その後に板寄せ方式で約定処理が実施される. 板寄せ方式は, 日本の市場で行われる一般的なバッヂオークションのメカニズムである. プレオープニング・セッションだ

けは、全てのエージェントが注文を投入することが可能な設定となっている。一方で、プレオーブニング・セッションの最後の板寄せによる約定後は、1ステップで1エージェントの注文しか受け付けない。100,500ステップの内訳は、100,000ステップはデータマイニングの学習用の期間であり、最後の500ステップが評価用のデータ期間として利用する。

トレーダーエージェントモデル

トレーダーエージェントのモデルとしては、先行研究[145]に基づいて作成された、Toriiら[53]のStylized Trader モデルを用いた。これは、4.3.2節で前述のモデルと基本的には一緒であるが、ここで再度説明する。

時刻 t において、Stylized Trader エージェント i は、ファンダメンタル、チャート(トレンド)、ノイズファクターに基づいて、注文行動の決定を行う。まず、エージェントは下記の通り3ファクターを計算する。

- ファンダメンタルファクター：

$$F_t^i = \frac{1}{\tau^{*i}} \ln \left\{ \frac{p_t^*}{p_t} \right\}. \quad (5.1)$$

ここで、 τ^{*i} はエージェント i の平均回帰時間のパラメータであり、 p_t^* は時刻 t におけるファンダメンタル価格、 p_t は時刻 t における価格である。

- チャート(トレンド) ファクター：

$$C_t^i = \frac{1}{\tau^i} \sum_{j=1}^{\tau^i} r_{(t-j)} = \frac{1}{\tau^i} \sum_{j=1}^{\tau^i} \ln \frac{p_{(t-j)}}{p_{(t-j-1)}}. \quad (5.2)$$

ここで、 τ^i はエージェント i のTime Window Size であり、 r_t は時刻 t におけるログリターンであり、価格時系列から計算可能である。

- ノイズファクター：

$$N_t^i \sim \mathcal{N}(0, \sigma). \quad (5.3)$$

ここで、 N_t^i は、平均 0、分散 $(\sigma)^2$ の正規分布を意味する。

続いて、エージェントは、上記の3つのファクターの重み付き平均を下記のように計算する。

$$\hat{r}_t^i = \frac{1}{w_F^i + w_C^i + w_N^i} (w_F^i F_t^i + w_C^i C_t^i + w_N^i N_t^i). \quad (5.4)$$

ここで、 w_F^i, w_C^i, w_N^i は、エージェント i の3つのファクターに対する重み付けである。

続いて、エージェント i の期待価格が以下の通り計算される。

$$\hat{p}_t^i = p_t \exp \left(\hat{r}_t^i \tau^i \right). \quad (5.5)$$

そのうえで、固定注文マージン $k^i \in [k_{\min}, k_{\max}]$ を用いて、最終的な注文は以下のように計算される。

- もし、 $\hat{p}_t^i > p_t$ であれば、エージェント i は以下の価格で買い注文を立てる。

$$\left\lfloor \min \left\{ \hat{p}_t^i(1 - k^i), p_t^{\text{ask}} \right\} \right\rfloor. \quad (5.6)$$

- もし、 $\hat{p}_t^i < p_t$ であれば、エージェント i は以下の価格で売り注文を立てる。

$$\left\lceil \max \left\{ \hat{p}_t^i(1 + k^i), p_t^{\text{bid}} \right\} \right\rceil. \quad (5.7)$$

ここで、 p_t^{bid} と p_t^{ask} は、売りと買いの裁量気配値を意味する。また、 $\lfloor \cdot \rfloor$ は床関数を、 $\lceil \cdot \rceil$ は天井関数を意味する。これらの関数は、呼値による価格の刻みをシミュレーションに反映するために適用している。

それぞれのパラメータは先行研究 [53] に基づいて設定している。 $w_C^i \sim Ex(1.0)$, $\tau^* \in [50, 100]$, and $\tau \in [100, 200]$ と設定した。おこで、 $Ex(\lambda)$ は、平均と分散が λ である指数分布である。なお、 w_F^i, w_C^i, σ については、実験で変更するため、後述する。

5.3.2 データマイニングモデル

人工市場の出力から、データマイニングモデルは、ザラ場の全ての注文データを入力として受け取る。これは、一般的な市場におけるティックデータ相当のもので、今回使用するデータマイニングモデルでは、以下の特徴量を入力として受け取る。

- 買いか売りか
- 価格
- 買いやの裁量気配値
- 売りの最良気配値

これらのシミュレーションから生成されるデータを、Train/Valid/Test の 3 つのデータセットに分割し、学習においては、Train と Valid を使用する。既に述べた通り、100,500 ステップ分のデータのうち、最後の 500 ステップ分は Test に使用する。そのため、データマイニングモデルの構築に使用可能なデータは 100,000 ステップ分になる。これを時系列の順に 9:1 に分けて、Train と Valid データとする。つまり、90,000 ステップ分のデータが Train データとなり、次の 10,000 ステップ分のデータが Valid データとなる。

予測タスクとしては、100 ステップ後の価格変動予測を過去の 32 注文時系列に基づいて行うというタスクを採用する。例えば、ステップ t において、入力は、 $(t-31)$ ステップ目から t ステップ目までのティックデータであり、予測ターゲットは $(t+100)$ ステップ目の価格である。予測タスクは 3 値分類タスクとした。つまり、 $(t+100)$ ステップ目の価格が t ステップの価格と比べて、Up/Stay/Down のどれであるかという予測タスクである。

上記の 4 つの特徴量からなる入力データは価格の絶対値による影響を軽減するためなどの理由から、以下の形式に変換される。

- 買いか売りか → 0/1
- 価格 → ファンダメンタル価格からの Tick 数¹
- 買いの裁量気配値 → ファンダメンタル価格からの Tick 数¹
- 売りの最良気配値 → ファンダメンタル価格からの Tick 数¹

データマイニングモデルとしては、アテンションベース、CNNベース、LSTMベースの3つのモデルを作成した。アテンションベースのモデルは、埋め込み層(ReLUアクティベーション層とLayer Normalizationを付した線形層)とアテンション層(Multi-head Attention)と合計層と2層の線形層(Layer Normalization適用済みで、1層目はReLUによるアクティベーション、2層目はSoftmax²によるアクティベーションを使用)からなるモデルである。CNNベースモデルは、1Dバッチノーマライゼーションとアベレージプーリングが適用された1D畳み込み層を3つ使用したのちに、2層の線形層(Layer Normalization適用済みで、1層目はReLUによるアクティベーション、2層目はSoftmax²によるアクティベーションを使用)からなるモデルである。LSTMベースモデルは、1層の埋め込み層(ReLUアクティベーション層とLayer Normalizationを付した線形層)とLSTM層を通ったのちに、2層の線形層(Layer Normalization適用済みで、1層目はReLUによるアクティベーション、2層目はSoftmax²によるアクティベーションを使用)からなるモデルである。

損失関数としてはCross-entropy Loss³を、OptimizerとしてはAdamを、Batch Sizeは128を、epochs数としては1000を、Learning Rateは 1.0×10^{-5} を採用した。これらの設定は、実験における学習の収束性の観点から設定した。

5.4 実験

実験は、主に、人工市場データマイニングプラットフォームとその実践的なモデルの一例がどのようにうまく機能するのかということを示すために行う。この実験では、人工市場シミュレーションのいくつかのパラメータを変更して、データマイニングの性能がどの程度変化するのかを評価する。これはまさに図5.1で示した通りの実験である。

この実験で変化させるパラメータは以下の通りである。

- シミュレーション内のエージェント数: $N_{agents} = 10, 30, 100, 1000$
- ファンダメンタルファクターの重み: $w_F^i \sim Ex(1.0), Ex(3.0), Ex(1.0)$
- チャートファクターの重み: $w_C^i \sim Ex(1.0), Ex(3.0), Ex(1.0)$

¹今回のシミュレーションの設定では、ティックサイズを1に、ファンダメンタル価格を500に設定しているので、価格から500を引いた値と同じである。

²実際にはlog Softmaxを用いて実装

³実際の実装にあたってはLog Softmaxとの相性の良さからNegative Log-likelihood Loss (NLL Loss)を使用している

- ノイズスケール: $\sigma = 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$

それぞれのパラメータの意味については 5.3 節を参照されたい。ここで, $Ex(\lambda)$ は, 平均と分散が λ である指数分布を意味する。便宜上, 以下では, $w_F^i \sim Ex(\lambda)$ を $w_F = \lambda$, $w_C^i \sim Ex(\lambda)$ を $w_C = \lambda$ と表記する。

それぞれのパラメータセットに対して, 異なる乱数シードを用いた 10 回の実験を行う。通常であれば, 正確な評価を行うためには 10 試行では足りないが, 以下の 2 つの理由から 10 試行に絞って実験を行った。一つ目の理由は, 計算リソースの限界である。それぞれの試行においては, 深層学習によるデータマイニング手法の学習も含むため, 非常に大きい計算リソースを必要とする。現在の設定では, $4 \times 3 \times 3 \times 5 = 180$ のパラメータセットが存在しており, それぞれのパラメータセットに対する試行数を絞らないと, 現実的な計算時間に収まらない。1 つのパラメータセットに対する試行数を 10 に絞ったとしても, 必要になるシミュレーション数は 1800 にもなる。第二に, シミュレーションが充分なステップ数あるということである。今回のシミュレーションでは, ファンダメンタル価格を 500 で固定している。そのため, とても離れた値を出すことはかなり限られている。無論, Fat Tail Property により, テイルの大きい価格分布をシミュレートする可能性は充分にあるが, それであっても, 平均回帰性などから, 比較的非定常性の度合いが低いと考えられる。そのため, 100,000 ステップというのはデータマイニングの観点からして, 充分なステップ数があり, 収束性が良いと考えられるからである。

データマイニング手法の評価手法として, 以下の指標を採用した。

- Accuracy
- Macro F1
- Micro F1
- Weighted F1

これらの評価指標はシミュレーションの最後の 500 ステップのテストデータから計算され, Valid データで最も良い性能を出したモデルを用いて計算される。

5.5 結果

3 つのモデルの各 1800 回のシミュレーション, 合計 5400 シミュレーションの結果を示す。ここではどのファクターが予測性能に大きな寄与度を持っているかについて分析を行う。全ての結果を示すことは困難であるため, ここでは, まとめた結果を示す。それぞれのファクターの影響は線形的なものではないと考えられるため, 結果の分析として, 評価指標の分布の密度を分析することで, その影響の分析を行う。

アテンションベース, CNN ベース, LSTM ベースの結果をそれぞれ, 図 5.3, 図 5.4, 図 5.5 に示す。それぞれの図では, 各列の左から右にそれぞれ, Accuracy, Macro F1, Micro

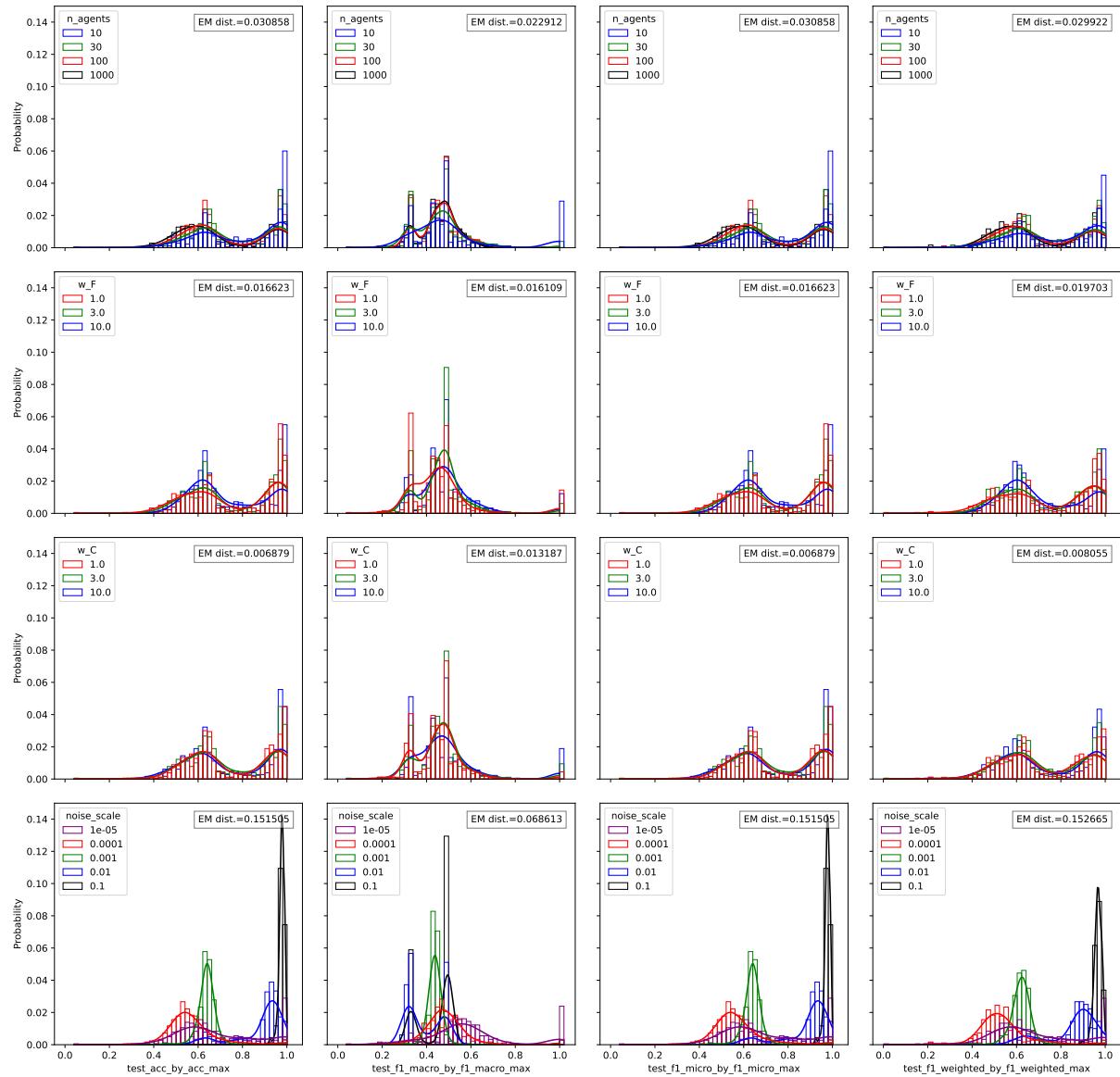


図 5.3: アテンションベースのモデルの結果

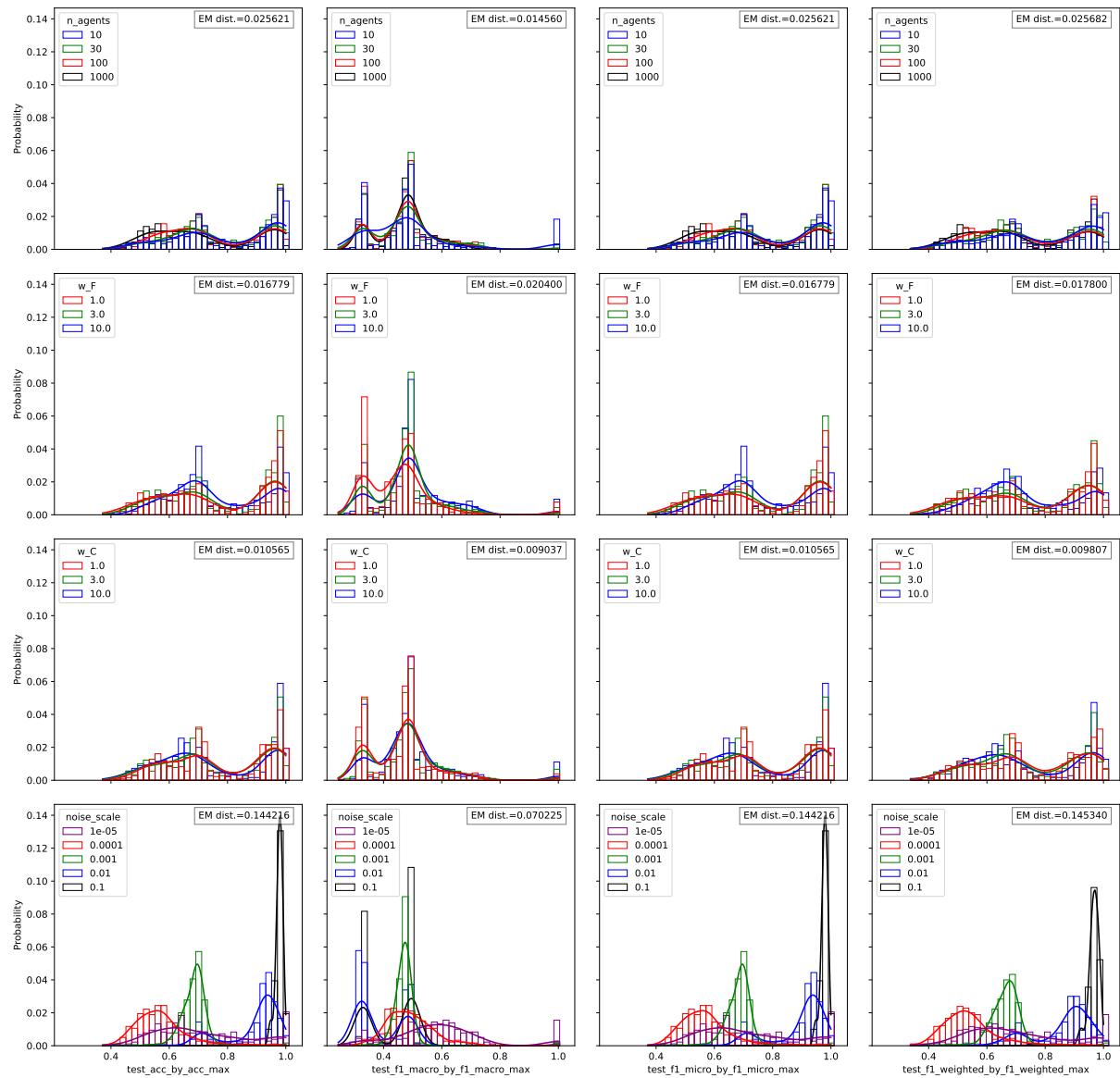


図 5.4: CNN ベースのモデルの結果

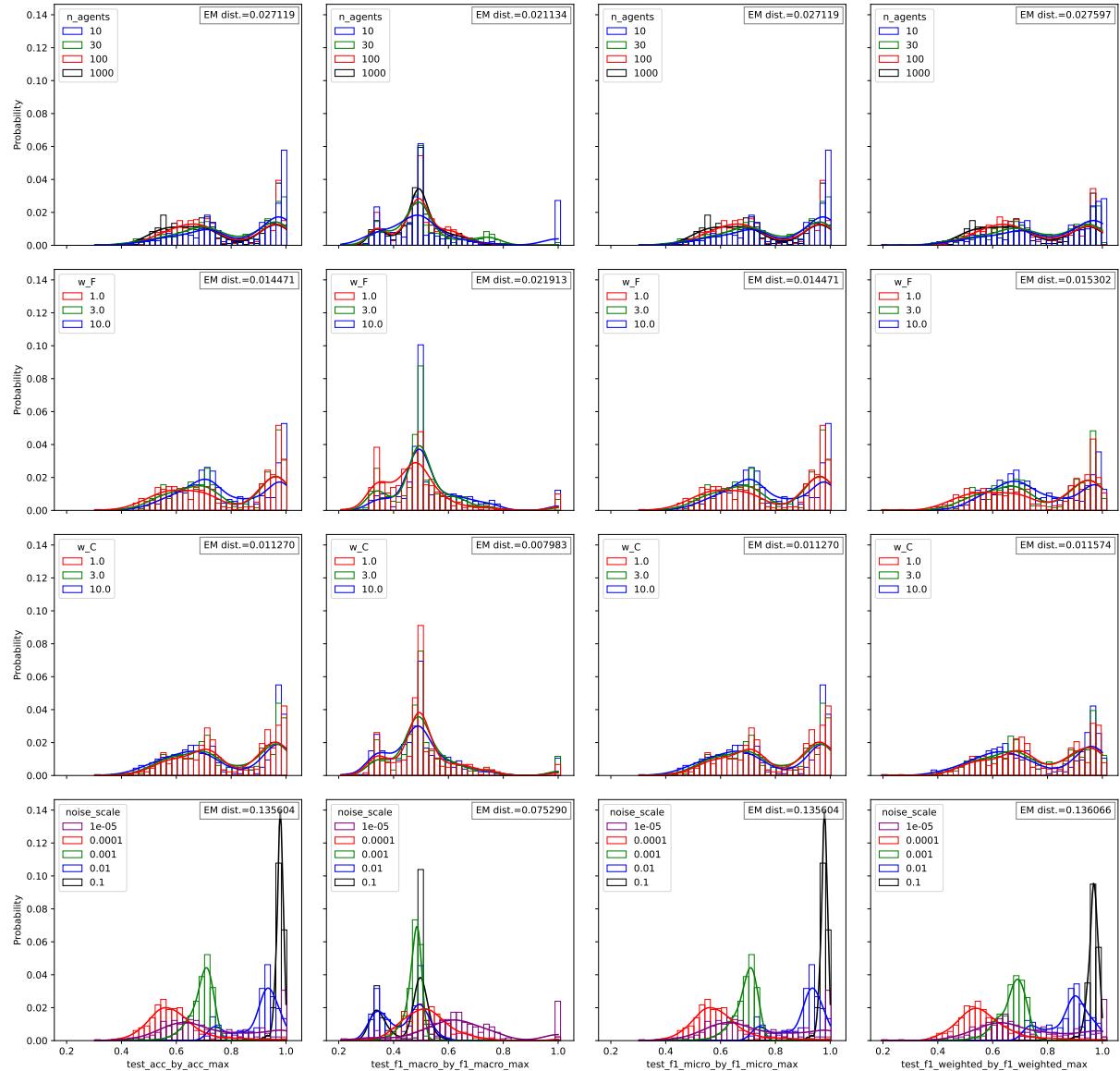


図 5.5: LSTM ベースのモデルの結果

F1, Weighted F1 の結果を示しており、各行は上から下に、 N_{agents} , w_F , w_C , σ ごとに集計をしたものであり、それぞれのヒストグラムは精度評価の分布を表している。また、各線は推定カーネル密度分布である。例えば、左下のプロットは、Accuracy に関して、ノイズスケールごとに分解して結果を集計したものであり、ヒストグラムはその分布を、線は推定カーネル密度を意味しており、それぞれの色は、ノイズスケール σ に対応している。つまり、簡単に言えば、左下のプロットは、Accuracy の分布に関して、ノイズスケールで分解した分析結果である。

さらに、それぞれのプロットで、EM (Earth Moving; Wasserstein) 距離の平均を計算したものが、各プロットの右上の角に表示されている。ここで、EM 距離の平均とは、全てのデータに対する評価指標の分布を計算した上で、それぞれのファクターで分解した後の評価指標の分布への距離を取るというものである。つまり、これは、各ファクターで分解した場合の、分解後の分布のばらつきを計算していることに相当しており、ばらつきが多いほど、そのファクターの分布の分解性能が高いことを意味する。そして、分解性能が高いほど、そのファクターは、予測タスクに対する影響が大きいということを意味する。そのため、EM 距離の平均が大きいほど、そのファクターの影響は大きいということになる。EM 距離は、非負値をとり、距離が 0 であることは、そのファクターは予測タスクに影響がないことを意味する。

結果の図と EM 距離の値から分かる通り、シミュレーションにおけるノイズスケールの影響が最も大きいことが言え、どの指標においても、他のファクターを変えた場合よりも明確に高い分解性能が確認できる。この傾向は、ヒストグラムからもカーネル密度からも観察できる。一方で、 N_{agents} , w_F , w_C を変えた場合（上から 1-3 行目）では明確な評価指標の分布の分解は明確には確認できない。

5.6 考察

本研究で変化させる対象とした 4 つのファクターのうち、ノイズスケール σ がデータマイニングの各種評価指標に対して最も高い影響性を持っていた。この傾向性は用いたデータマイニングの手法に拘らず確認できる傾向であった。他の要素の影響は否定できないものの、図 5.3, 5.4, 5.5 を見る限りでは、明確にそれらの分解能と比べると、ノイズスケール σ の分解能は顕著であった。

ノイズスケールの影響が最も大きいという結果は、比較的自然な結果であると考えられる。今回のシミュレーションにおいて、ノイズスケールは、トレーダーのパラメータとしてモデリングされている。しかしながら、このノイズは様々なノイズを含んでおり、観測ノイズや、ファンダメンタル価格の推定ノイズ、人間の行動ノイズなどを包含する形でモデリングされている。また、実際の市場における予測困難な価格変化のようなノイズは、多くのトレーダーの行動の創発現象として発生する。そのため、このシミュレーションにおいては、ノイズを細かく定義してモデリングすることは困難であることから、トレーダーのパラメータの一部としてモデル化し、トレーダーの行動を通じてマーケット全体の

ノイズとして機能するように設計している。そのため、データマイニングの観点からすると、このノイズの大きさが変化することで、予測の難しさが高まることは、自然なことである。

一方で、ノイズスケールの影響は単純ではないこともわかった。結果によれば、ノイズスケールが 0.001 より小さい場合に、評価指標が低い結果となっており、難しいシチュエーションであるということが分かる。一方で、ノイズスケールが 0.1 や 0.01 である時に、評価指標が非現実的なレベルで高くなっていることが確認できる。これは、非常に大きいノイズの場合は、トレーダ自身が価格予測が難しいために、一方向に注文が偏ることが発生しない状況となり、Stay のクラスになることが頻発することで、予測が簡単になるためであると考えられる。これは、Macro F1 に関しては、評価値が高くなっていないことと整合性がとれる結果となっている。一方で、程よいサイズのノイズスケールの際に予測が難しくなっているということも確認できた。実際に、先行研究 [53] では、Stylized Facts の再現できるパラメータとして、ノイズスケールを 0.001 に設定しており、これとの整合性もある結果となった。

これらの結果を踏まえて考えると、今回の分析で得られた結果は、妥当性のある結果となっており、人工市場データマイニングプラットフォームの有用性と妥当性を確認できる結果となっていると考えている。得られた結果は、ある程度事前に予想可能である内容であったものの、パラメータごとの価格予測難易度への寄与度の違いを明確に確認することができている。そのため、金融市場という複雑システムにおけるデータマイニングの性能評価を様々なシチュエーションごとに比較する手法としての有用性は認めることができると考える。

本章では、人工市場のパラメータのうち、4つのパラメータのみを変更して分析を行ったが、実際には、もっと他の人工市場のパラメータや要因を変化させて、予測精度に与える影響を検証することが可能であると考えている。すでに述べた通り、人工市場の設定は変更可能である。そのため、本章の実験とどうように、検証したい点に合わせて様々なものを変化させ、評価を行うことができる。たとえば、人工市場の設定を変えることさえ也可能である。世界的には、様々な市場が存在し、様々な取引ルールで、様々な種類のものが取引されている。そのため、人工市場のモデルさえうまく変更すれば、本章で提案しているコンセプトを適用して、様々な比較を行うことができると考える。特に、本研究で使用している人工市場シミュレーションプラットフォームの PlhamJ [58] は、活用の幅が広く、電力市場に拡張して実験を行った研究 [146, 147] も存在している。そのため、本章で提案している人工市場データマイニングプラットフォームのコンセプトの適用範囲はまだまだ広げることができると考えている。

また、本章では明確な比較実験を行ってはいないものの、データマイニング手法間で比較のための、ベンチマーク用プラットフォームとして使うなどの別の応用も考えられる。本章では、人工市場における各パラメータの影響にのみ着目し、各データマイニング手法の違いについては考察を行わなかった。これは、本章で採用したデータマイニングのモデルがどれも非常に基本的なものであり、それらの間での結果の差がほとんどないため

である。しかしながら、このプラットフォームは、データマイニング手法の評価プラットフォームとして利用できると想定している。人工市場シミュレーションは完全に我々がコントロール可能であり、実際の市場のデータがなくともデータを生成可能である。そのため、パラメータを変更して、様々な状況下での評価を公平に行うことが可能であると考えている。また、II部で提案したような、実データを用いたより現実的な人工市場モデルの構築を行えば、より現実的な評価プラットフォームを構築できると考える。

また、本章での結果は、提案プラットフォームが人工市場シミュレーションのパラメータチューニングにも利用できる可能性を示唆している。すでに考察した通り、ノイズの規模が非現実的なレベルで大きい場合に、データマイニングの予測精度が現実的でないほど高くなっている。そのため、逆に、データマイニング手法が非現実的なレベルで高い予測性能を示さないということが、人工市場のパラメータを絞り込むことに利用できる可能性を示している。

今後の課題としては、さらなる実用例を増やし、様々な分析を行うことや、この提案プラットフォームの有用性のさらなる確認などがあげられる。

5.7 本章のまとめ

本章では、人工市場シミュレーションとデータマイニングを組み合わせた、人工市場データマイニングプラットフォームのコンセプトを提案した。このプラットフォームの利点は、人工市場がすべてコントロール・観測可能であることであり、これを活用して、人工市場のパラメータやファクターを変更して、市場において支配的なパラメータを見つけて、様々な状況を検証したりできる。この人工市場の利点に対して、データマイニングの手法を組み合わせることで、金融市場におけるデータマイニングのモデルを実データを用いることなく、公平に評価できることを可能になると考えている。この評価を通じて、金融市場の予測困難性を生み出す要素の発見も可能であると考えている。

本章では、このプラットフォームのコンセプトの提案を行うとともに、その妥当性を確認するための実験を行った。実例として、ティックタイムレベルの人工市場シミュレーションと、短期の価格変動予測のデータマイニングモデルを用いて、4つの金融市場のパラメータがどのように価格予測の困難性を生んでいるかを検証した。その結果、金融市場のノイズスケールがデータマイニングの性能に大きく寄与しており、複雑な効果が確認された。

これらの実験を通じ、人工市場データマイニングプラットフォームの一定の妥当性と有効性を確認できた。

次章では、実際にこの人工市場データマイニングプラットフォームを用いて、別の分析を実施することで、金融市場の予測困難性に関する分析を行う。

第6章 ケーススタディー：人工市場による注文同時性の効果分析—金融市場における注文の同時性はデータマイニングタスクに影響を与えるか？

6.1 研究背景：金融市場の高速化と注文の同時性

情報処理技術の向上とともに、金融市場の取引速度も向上してきている。高頻度取引の市場シェアが縮小傾向にあることも指摘されている [148] ものの、現在も、世界中で注文の時間優先の原則は取引ルールに取り込まれており、高頻度取引という手法自体はいまだに有効な手段であると考えられる。これらの高速化は、今後も技術発展とともに上昇していくことが想定される。

高頻度取引の出現とともに、その長所・短所は議論が繰り広げられてきた。例えば、高頻度取引の存在は、フラッシュクラッシュを引き起こす可能性が指摘されている。一方で、高頻度取引が存在することにより、短期的な市場の価格発見機能が高まるというメリットもあげられる。

高頻度取引の短所に対応するために、スピードバンプも提案されている。スピードバンプとは、金融市場において、注文処理を遅らせるルールを追加することで、高頻度取引の速度を抑えようという取り組みである。スピードバンプは、2017年にNYSE Americanで導入されたことがある。さらに、欧州各国でも、MiFID II や HFT Act などで検討が行われている。また、他の高頻度取引への対処として、すべての注文処理をバッチオーダーショップ方式にする方法なども存在する。

このような、高頻度取引の高速化とスピードバンプに関連して、注文の同時性という概念がとても重要である。注文を出す際に、注文のネットワーク伝送中や処理中に同時に他の注文も出される可能性があるため、注文判断を行う際に参照できる情報は、その注文の直前に処理される注文を含まない可能性が高い。もし、スピードバンプが導入されると、この同時性が高まる。一方で、市場の高速化が進んだ場合には、高速化により、自分の注文の投入から処理の間に注文が出される確率が低くなるため、同時性が低下する。こ

の注文の同時性は、取引中の市場の不確実性を高め、情報の遅延を相対的に大きくするため、高頻度取引の敵となる存在である。そのため、高頻度取引規制は注文の同時性と密接な関係にあるといえる。究極的には、全ての注文が同時に処理されるバッヂオーケーションが最も注文の同時性が高いということになる。

さらに、注文の同時性は、短期間での注文順序の意味をなくす。非常に短い期間に出される連続する注文は、異なる順番で到達する可能性は十分にあり、その順序に必然性はない可能性が高い。さらに、前述の通り、短い間隔の注文同士が双方の存在を考慮に入れているとは考えづらい。例えば、ある注文の100ミリ秒後に、別の注文が処理されたとしても、後者の注文の発注時に、前者の注文を考慮に入れて注文を出しているとは考えにくい。

また、注文の同時性は、金融データマイニングにも影響が及ぶと考えられる。近年、GANによるデータ拡張手法が提案されており、その利点が指摘されている[107]。本論文でも3章で実際にGANを作成している。GANは、通常、注文時系列が与えられ、次の注文を生成するというのが一般的である。これを繰り返すことで、より長い注文時系列を生成するのが一般である。

一方で、このような注文時系列の生成は、注文の同時性を仮定した場合に、意味があるのであろうか？もし、注文の同時性が短期間での注文の順序を無意味にすると仮定するのであれば、次の注文を予測/生成使用用とする取り組みは無意味に思える。なぜなら、複数の注文の到着におけるランダム性が次の注文の予測に大きく影響を及ぼし、全く予測できない可能性があるからである。

そこで、本章では、マルチエージェントシミュレーションを用いて、注文の同時性を仮定したときに、GANによる注文の生成に意味があるのかについて、分析を行う。つまり、注文の同時性が次の注文の生成(予測)タスクの精度に大きく影響するという仮説の検証を行う。実験に当たっては、前章の人工市場データマイニングプラットフォームを活用する。また、GANのモデルとしては、3章のPolicy Gradient Stock GAN (PGSGAN)を用いる。

6.2 モデル

モデルは5章で提案した、人工市場データマイニングプラットフォームを用いる。

概要を図6.1に示す。人工市場シミュレーションは、他の章と同様に、先行研究[53]をベースとしたシミュレーションを用いる。このシミュレーションには、1つの連続ダブルオークション市場と市場データを参照しながら取引を行うトレーダーエージェントが存在する。人工市場シミュレーションを用いることで、通常では観測できない注文者情報を特定できるため、注文の同時性の効果が検証可能になる。また、人工市場シミュレーションでは、市場環境を完全にコントロール可能であるため、様々な状況の検証が可能である。

注文は、ティックデータとして集約され、データマイニングに使用される。本研究では、データマイニングモデルとして、3章で提案したPGSGANを用いる。そして、そのGANモデルの構築を、すべての注文を対象に行うケースと、注文者別に行うケースで

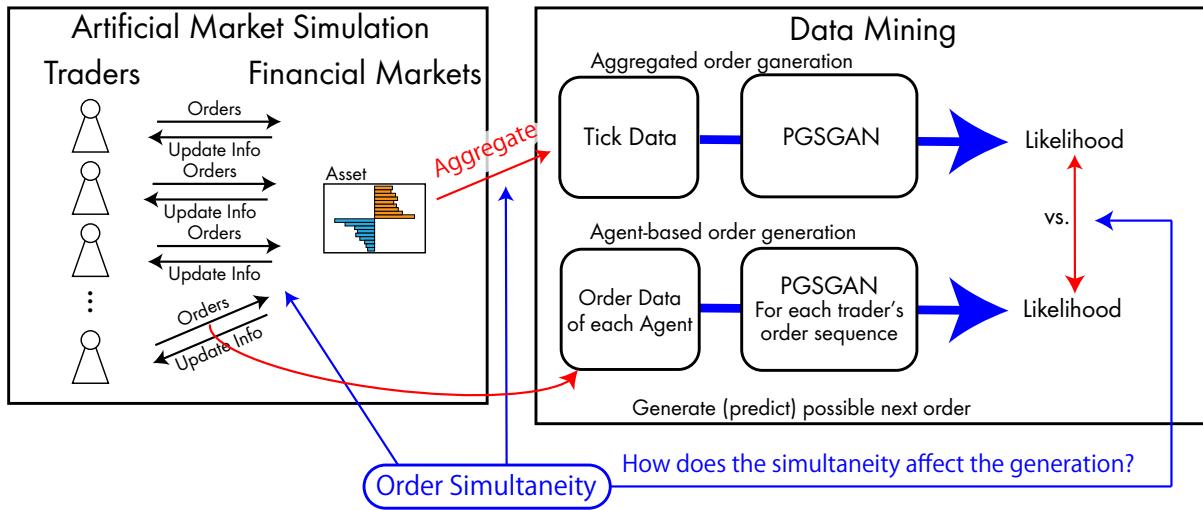


図 6.1: 本章における、人工市場データマイニングプラットフォームの詳細

わけて、その精度比較を行うことで、注文の同時性の影響を検証する。ここで、PGSGANを採用したのは、PGSGANでは、注文生成に当たって、Policyの作成を行うために、そのPolicyを活用すると、注文生成の分布とデータを突き合わせることで、尤度を計算可能であるためである。尤度計算が可能になると、どちらのモデルの方がどの程度当てはまりがよいのかということを定量的に比較可能になる。

以下では、これらのモデルの詳細を説明する。

6.2.1 人工市場シミュレーションモデル

個別エージェントの行動プロセス

前章の 5.3.1 節で説明したものと同じ行動プロセスを持つエージェントを用いた。前章同様に、 w_F^i, w_C^i, σ については、実験で変更するため、後述する。

マーケットの注文の処理

各ステップで、エージェントが注文を出すが、本章の実験だけは、他の章と異なり、注文が処理される前に、すべての注文の順番をランダム化する。この操作により、注文の同時性を仮想的に実現する。

6.2.2 データマイニングモデル: PGSGAN

データマイニングモデルとしては、3章で提案したPGSGANを用いる。

ただし、シミュレーションの出力に合わせるために、一部の変更を行った。まず、注文情報の入力を変更した。従来の実装では以下の7つの情報が入力に含まれていた。

- 売りか買いか？
- 新規注文か？
- 成行注文であるか？(Is MO)
- 相対価格 (最良気配値からの Tick 数)
- 注文数量 (最小注文単位で割った数値)
- 現在の売りの最良気配値 (相対値)
- 現在の買いの裁量気配値 (相対値)

一方で、シミュレーションにおいては簡便化のため、キャンセル注文と指値注文を考慮に入れていないうえ、数量については1に固定されているため、不要な次元が存在する。そのため、本章の実験では、以下の通りに変更した。

- 売りか買いか？
- 相対価格 (最良気配値からの Tick 数)
- 現在の売りの最良気配値 (相対値)
- 現在の買いの裁量気配値 (相対値)

この変更に対応して、PGSGAN が生成する注文の特徴量も変更した。従来の実装では、

- 売りか買いか？ – 2 クラス (確率値)
- 新規注文か？ – 2 クラス (確率値)
- 成行注文であるか？(Is MO) – 2 クラス (確率値)
- 相対価格 (最良気配値からの Tick 数) – 40 クラス (確率値)
- 注文数量 (最小注文単位で割った数値) – 40 クラス (確率値)

一方で、本章の実験においては以下の通りに変更した。

- 売りか買いか？ – 2 クラス (確率値)
- 相対価格 (最良気配値からの Tick 数) – 40 クラス (確率値)

加えて、従来のモデルでは、20注文時系列を入力として採用していたが、本章のモデルでは、32注文時系列を入力として採用した。これらの変更に対応して、PGSGAN の Generator と Critic の層の数も変更した。また、他の詳細については基本的には3章と同じであり、本章では、Hinge Loss を採用しているモデルを用いた。

6.2.3 仮説と検証手法

本章においては、注文の同時性が次の注文生成（予測）タスクの精度に大きく影響すると仮定した。ここで、説明の簡便化のために、この注文の同時性が10であるとする。（実験では10体のエージェントを同時に動かすことで同時性を実現している。）簡単に言えば、常に10個の注文がほぼ同時に発注される状況を考えることになる。ここで、 x_i を*i*番目に市場に到達した注文とします。 x_i から x_{i+9} の注文が同時的に発注されたとすると、以下の条件付き確率式が成立すると考えることができる。

$$P(x_i|x_{i-1}, x_{i-2}, \dots, x_0) = P(x_{i+1}|x_{i-1}, x_{i-2}, \dots, x_0) = \dots \quad (6.1)$$

$$= P(x_{i+9}|x_{i-1}, x_{i-2}, \dots, x_0). \quad (6.2)$$

つまり、 x_i から x_{i+9} の注文の同時性が仮定できるとすると、それらのどれが x_{i-1} の次の注文として登場するのかということに関しては、等価に起こり得る事象ということになる。あくまで、 x_i は、実際に実現した次の注文にすぎず、仮説に基づけば、確率論的には $x_{i+9}, x_{i+8}, \dots, x_i$ のどれが来てもおかしくないということである。今回ターゲットしている注文の生成タスクは $P(x_i|x_{i-1}, x_{i-2}, \dots, x_0)$ の推定そのものであり、注文の同時性を仮定すると、このタスクは難しくなるということが言える。

そして、PGGANにおいては、 $P(x_i|x_{i-1}, x_{i-2}, \dots, x_{i-32})$ そのものを間接的に推定していることになる。そのため、もし、PGGANが正確に次の注文を生成できるのであれば、生成された確率分布であるポリシーにおいて、実際に実現した注文 x_i に対する尤度 $P_{G(z)}(x_i)$ は高くなるはずである。そのため、 $P_{G(z)}(x_i)$ を計算することで、生成の精度を調べることが可能になり、この調査を通じて、注文の同時性の影響を確認することができるはずである。

仮説検証に当たっては、比較実験を行った。その概要是図6.1に示されている通りであるが、2つの実験の精度の比較を行うことで仮説検証を行っている。

1. Aggregated order generation (AggOG): 単純に次の注文を生成するタスク
2. Agent-based order generation (ABOG): 各エージェントごとに、それぞれのエージェントが次に出す注文を生成するタスク

後者のタスクにおいては、注文の同時性によるランダム性を排除することが可能である。また、後者のタスクにおいては、各エージェントに対してGANのモデルが作成されるため、これらに基づく平均尤度を計算することで、前者のタスクとの比較を行うこととする。

具体的には評価指標として、平均NLL、つまり、 $NLL_{G(z)}(x)$ を採用する。ここで、 x は、シミュレーションにおける次の注文である。NLLは低いほどうまく予測できていることを意味し、NLLは0以上の値をとるため、AggOBのNLLがABOGのNLLを上回れば、注文の同時性の影響により、予測精度が下がっていることを意味し、本章における仮説を棄却できないことになる。そこで、AggOBのNLLとABOGのNLLの値の差に関して、*t*検定を行い、*p*値も計算することで、統計的な分析を行う。

6.3 実験

実験として、人工市場のパラメータのいくつかを変更して、様々な環境における、注文の同時性の影響を分析した。変化させたパラメータは以下のとおりである。

- ファンダメンタルファクターの重み： $w_F = 3.0, 10.0.$
 - チャートファクターの重み： $w_C = 3.0, 10.0.$
 - ノイズスケール： $\sigma = 10^{-2}, 10^{-3}, 10^{-4}.$
- その他の固定パラメータは以下のとおりである。
- 人工市場シミュレーション：
 - ノイズファクターの重み： $w_N = 1.0.$
 - エージェント数：10 (同時性の数)
 - 開始価格：500
 - ファンダメンタル価格： $p_t^* = 500$ (固定)
 - 呼値：1
 - 寄付前ステップ数：100 steps
 - ザラ場のステップ数：10500 steps
 - PGSGAN：
 - 学習エポック数：1000
 - 学習率：0.001
 - Two Time-scale Update Rule [141]: G:C=1:5
 - 価格クラス：20 (最良気配値から20Tick以上離れた場合は20としてカウントする)
 - ヒストリカルデータの入力：32 注文
 - Generatorに使用する乱数の次元数：10
 - 学習データ：シミュレーションで生成されたデータの冒頭10000ステップ分
 - テストデータ：シミュレーションで生成されたデータの最後500ステップ分

各パラメータセットに対して、異なるシミュレーションの乱数シードを用いて100回のシミュレーションを行い、そのデータに対する、AggOGとABOGのPGSGANのNLLを比較した。NLLの計算に当たっては、生成されたポリシーのGANへの入力乱数によるばらつきの影響を軽減するために、各評価で、PGSGANに100個の異なる乱数を入力し、100回の評価の平均 w のとることで、平滑化を図って評価を行った。

表 6.1: AggOG と ABOG の NLL の評価値の一覧

w_F	w_C	σ	NLL (AggOG)	NLL (ABOG)	diff	t-value	p
10.0	10.0	10^{-4}	2.2415 ± 0.0465	1.0064 ± 0.0283	1.2352	229.2650	$\ll 0.01$
10.0	3.0	10^{-4}	2.3222 ± 0.0424	1.1098 ± 0.0211	1.2123	258.5539	$\ll 0.01$
3.0	10.0	10^{-4}	2.2354 ± 0.0491	1.1030 ± 0.0287	1.1324	200.9167	$\ll 0.01$
3.0	3.0	10^{-4}	2.3832 ± 0.0474	1.2646 ± 0.0242	1.1186	212.2086	$\ll 0.01$
10.0	10.0	10^{-3}	2.5712 ± 0.0407	1.7460 ± 0.0309	0.8251	163.0315	$\ll 0.01$
10.0	3.0	10^{-3}	2.6721 ± 0.0354	1.9832 ± 0.0315	0.6889	146.8755	$\ll 0.01$
3.0	10.0	10^{-3}	2.6388 ± 0.0415	1.9369 ± 0.0344	0.7019	131.6461	$\ll 0.01$
3.0	3.0	10^{-3}	2.7304 ± 0.0239	2.2019 ± 0.0269	0.5285	148.6249	$\ll 0.01$
10.0	10.0	10^{-2}	1.7224 ± 0.0411	1.7036 ± 0.0337	0.0188	3.5676	$\ll 0.01$
10.0	3.0	10^{-2}	1.4544 ± 0.0332	1.4768 ± 0.0305	-0.0225	-5.0303	$\ll 0.01$
3.0	10.0	10^{-2}	1.1071 ± 0.0323	1.1713 ± 0.0321	-0.0641	-14.2390	$\ll 0.01$
3.0	3.0	10^{-2}	1.0898 ± 0.0070	1.1432 ± 0.0092	-0.0535	-46.6007	$\ll 0.01$

6.4 結果

表 6.1 と図 6.2 は、すべての結果を示している。NLL (AggOG) と NLL (ABOG) は、AggOG と ABOG の NLL の平均値を示しており、値が小さいほど PGSGAN が次の注文生成に成功していることを示している。加えて、表 6.1 における “diff” は、NLL (AggOG) と NLL (ABOG) の値の差を示しており、 $NLL(\text{AggOG}) - NLL(\text{ABOG})$ で計算されている。“diff” が負の値をとる時、AggOG の推定の方が ABOG よりも良いこととなり、本章で検証している仮説に反することとなる。統計的検定として、NLL (AggOG) と NLL (ABOG) の差に関する t 値と p 値も計算した。

結果によれば、AggOG でも OBOG でも GAN がそれなりに次の注文生成に成功していることが分かった。これは、GAN の生成ターゲットが売り/買いの 2class 分類と 20 クラスの相対価格の予測からなるため、NLL のチャンスレートが $\ln\{2 \times 20\} \approx 3.6889$ であり、これよりも十分に低い NLL となっているためである。

表 6.1 によれば、ノイズスケール σ の影響が最も支配的であり、 w_F や w_C の影響より断然大きいことが分かった。また、ノイズスケールが小さい場合には、AggOG と ABOG の差が大きいことがわかる。一方で、ノイズスケールが大きい場合には、この差がなくなることがわかる。さらに、興味深いことに、ノイズスケールを $\sigma = 10^{-2}$ に設定した場合、いくつかのパラメータセットにおいて、ABOG の NLL が AggOG の NLL を上回っており、各エージェント個別の注文の生成よりも、すべてを集約した注文の同時性の影響のあるはずの注文時系列のほうがより正確に次の注文を生成できていることが示された。この結果は、本章で検証している仮説に反する結果である。

また、 w_F と w_C の効果を比較すると、 w_F の方が w_C よりも大きい効果を持っていることがわかった。ただし、前述の通り、ノイズスケールと比較するとその効果は小さい。

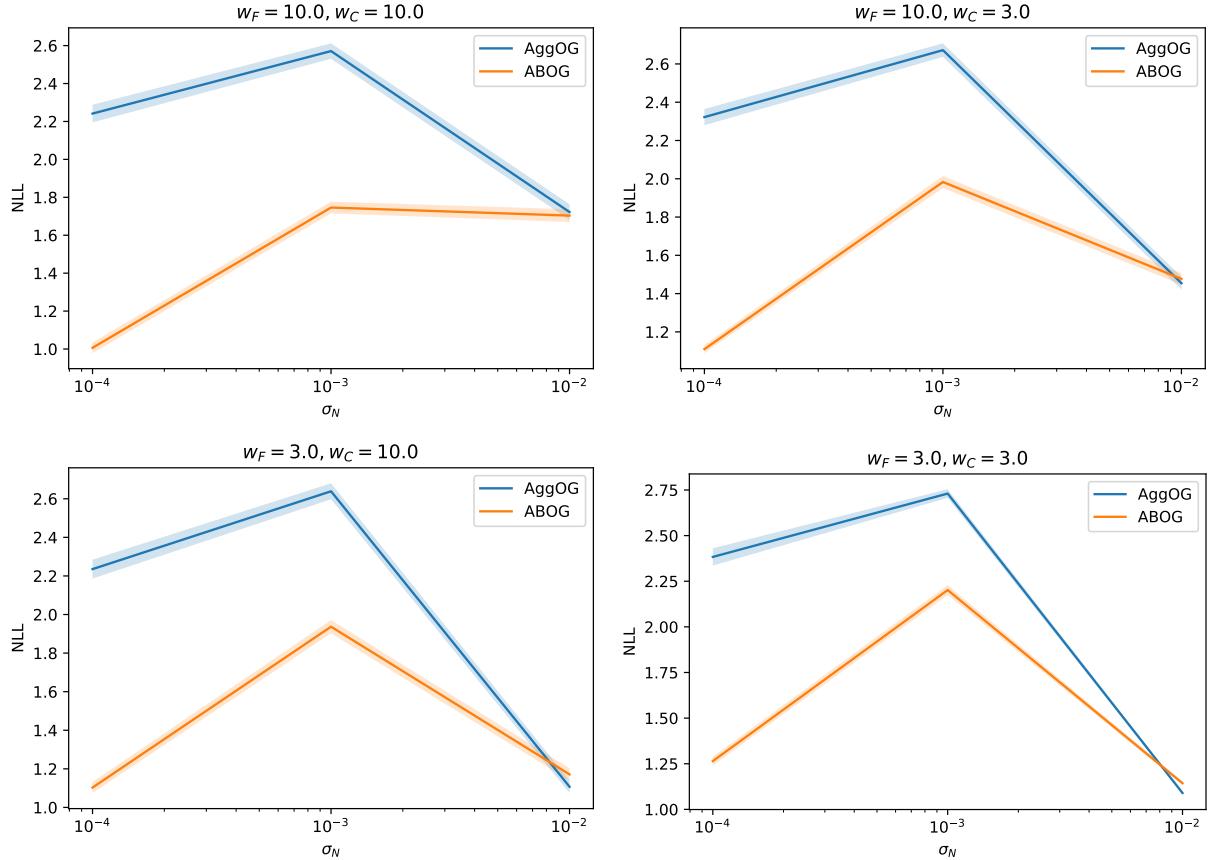


図 6.2: すべてのパラメータセットの結果のグラフ. 青いラインは AggOG の NLL を示しており, オレンジのラインは ABOG の NLL を示している. 塗りつぶしエリアは標準偏差を示している.

6.5 考察

まず, 本章で得られた結果は, 予測(生成)タスクの難易度と市場のノイズスケールの関係を示していると言える. 興味深いことに, 中程度のノイズスケール(本研究では $\sigma = 10^{-3}$ 程度)が最も高い NLL を示し, 予測精度が最も低く, 難易度が最も高いことを示している. この現象の理由は以下の 2 つにあると考えている. 一つは, ノイズスケールの直接的な効果である. ノイズが大きいと予測が難しくなるというのは, ノイズスケールの一般的な効果であると捉えられる. もう一つは, エージェントの行動を介したノイズスケールの間接的な効果である. ノイズスケールが大きくなると, エージェントは市場の方向性を明確にとらえることができなくなり, 多くのエージェントが方向性のない似たような注文を出すようになる. その結果として, すべての注文が似たようなものとなり, 予測が簡単になるという状況である. この二つの要素が複雑に重なり, 中程度のノイズの場合に予測が困難になるのではないだろうか. また, ノイズスケールの現実性という観点で行くと, 先行研究 [53] では, パラメータの実市場へのフィッティングの結果として, $\sigma = 10^{-3}$ という

バラメータを採用している。そのため、実市場における予測の難易度の上記のような現象が発生している可能性が大いにある。

続いて、注文の同時性の影響は、ノイズスケールが小さいほど大きくなつた。これは、表6.1における“diff”で記載されている NLL (AggOG) と NLL (ABOG) に差から観測できる。ノイズのサイズが小さい場合には、エージェントは自分の注文の方向性を明確に決めることができるために、各エージェントの注文分布の重なりが小さくなり、注文の同時性ら生じるランダム性の効果が高まる。一方で、ノイズサイズが大きく、エージェントの行動の方向性が明確ではない場合には、方向性の明確ではない似たような注文ばかりが出るため、各エージェントの行動の分布の重なりが大きくなり、注文の同時性の影響が小さくなると考えられる。

ここまでで、ノイズに関して分析を行つたが、本研究でモデル化したノイズには多くの種類のノイズが含まれている。ノイズは、式5.3で定義されており、式5.4を通じて、エージェントの行動の一部としてモデル化されている。Stylized Trader のモデル化では、ノイズは単なる正規分布としてモデル化されているが、実際の市場におけるノイズの意味は幅広く、それらを実質的に包含している。たとえば、各エージェントのファンダメンタルに関する観測ノイズ、各エージェントの推定誤差、外部情報に関する観測ノイズ、各エージェントの行動決定における曖昧性、市場の不確実性など、さまざまな要素がこのノイズで一括でモデル化されている。また、実際の金融市場においては、ノイズは、イベントや個別の株主の行動の積み重ねとして発生するが、本研究で使用しているモデルでは、シミュレーションの簡便のため、それらのノイズをまとめて各エージェント内部の1つの確率変数でモデル化している。このような簡便化されたノイズのモデル化を通じて実際の市場の変動をシミュレーションしているが、これらのモデリングでは、Stylized Facts が再現できることから、妥当性があると考えられる。

続いて、金融市場の注文の同時性が、次の注文の予測/生成精度に大きく影響を与えるという本章における仮説について検討を行う。まず、本章で、GAN による次の注文生成をタスクとして採用したのは、このタスクが完全に実現可能である場合、次の注文の生成を繰り返すことで、注文時系列全体の生成が容易であるということになるため、究極の予測タスクと言えるからである。GAN は通常、生成にランダム性があるが、生成におけるランダム性をサンプリング回数を増やすことで平滑化すれば、究極の予測タスクとしてみなすことが可能である。さらに、PGSGAN により、注文分布を用いて注文生成を行うことが可能になったことにより、より正確な尤度の計算が可能になった。

結果に基づけば、一部の条件下では、注文の同時性の効果は、部分的に認められ、AggOG と ABOG の NLL の差が 0 であるということを統計的に棄却できる。そのため、注文の同時性により、生成タスクが困難になることがあるということは否定できない結果となった。

一方で、注文の同時性の効果が確認できる場合であっても、その効果は限定的であると言える。本章の実験では、NLL の差は、1.24 以下である。これは、 $\exp(1.24) \approx 3.46$ と計算すれば、同時性を認められる範囲での注文のバラエティーは 3.46 パターン程度であることになる。本章の実験では、注文の同時性を 10 と仮定して実験を行つたため、こ

の，3.46という数字は，10よりは断然低い値ということになる。これは，注文の同時性の効果は限定的であることを意味する。

加えて，チャンスレートと比較しても，注文の同時性は，次の注文の生成に与える影響は限定的である。すでに計算を示したが，チャンスレートのNLLは3.6889である。しかし，注文の同時性を仮定しても(AggOG)，NLLはチャンスレートよりも大幅に下回っている。このことも，注文の同時性の効果が限定的であることを裏付けていると言える。

さらに興味深いときに， $\sigma = 10^{-2}$ の場合に，AggOGにおいて，ABOGよりも高い尤度が確認できるケースが発生している。これは，注文の同時性に関する仮説には相反する，想定の逆効果が発生している。これは，予測難易度によるものではないかと考えている。つまり，ノイズが大きい場合には，前述の通り，予測が難しくなる。一方で，注文が集約されて1つのモデルだけを作成するAggOGにおいては，ABOGに比べると，注文が集約されている分，1つのモデルの作成に使用できるデータの量が圧倒的に大きくなる。そのため，予測が難しい環境下においては，データが多い方が有利な結果になっているのではないかと考えている。特に，金融市場においては，実現された価格の変動のパスは，起り得る価格変動のパスの空間のうちの1つでしかなく，非定常性などの問題により，データの不足が起きやすい。そのため，個々のエージェントごとに分割されない場合の方が，充分なデータを確保できる分，有利である可能性が考えられる。

さらに，注文の同時性効果が限定的であることは，注文生成を集約された注文時系列として生成を行うことの合理性を示しているとも考えられる。金融市場における合理的期待仮説によれば，市場全体の動きは，市場参加者の期待価格として実現される。そのため，集約された注文時系列を生成すること自体が，この合理的期待仮説と一致した方針であると言える。このことは，個別エージェントごとに注文予測をするということが，市場の動向予測には無意味である可能性を示唆しているとも考えられる。

これらの分析に基づけば，注文の同時性が大きい場合でも，集約されたデータを用いるGANのアプローチは，金融市場のモデル化において，ある程度の合理性があるということが分かった。また，金融市場のデータ拡張手法という点でいえば，人工市場シミュレーションのような個々のトレーダーをモデル化しようとするアプローチは，本研究のような仮説検証研究のための利用以外で実用上の限界があるため，市場の行動を包括的のモデル化するGANのようなアプローチが一定程度合理的であると言える。

最後に，人工市場データマイニングプラットフォームの活用に関して考察する。本章では，前章で提案した，人工市場データマイニングプラットフォームを活用して，複雑な現象を分析し，説明性のある結果を得ることができた。これは，まさに人工市場データマイニングプラットフォームの利点であり，このプラットフォームが有効であることを示していると考えている。今後の課題として，このプラットフォームを用いたさらなる仮説検証が考えられる。また，この人工市場データマイニングプラットフォームは，金融市場だけでなく，他の創発的な現象の起こる，複数行動主体による現象にたいしても有用である可能性が高いと考えられる。

6.6 本章のまとめ

本章では、金融市場における注文の同時性に着目した。金融市場では、注文がほぼ同時に市場に発注されることが多くあり、トレーダーはそれらの注文を考慮に入れて注文を出すことができない。本章では、このような現象を、注文の同時性と呼び、データマイニングタスクへの影響を分析した。データマイニングのタスクとしては、GANによる次の注文の生成を採用した。これは、注文の同時性が高まった場合に、次の注文のランダム性が高まるため、注文の同時性とGANによる次の注文予測は相反するとも考えられるからである。そこで、本章では、前章で提案した、人工市場データマイニングプラットフォームのアイデアを活用し、完全にコントロールされた環境下で、注文の同時性がGANの性能に与える影響を評価した。その結果、注文の同時性の影響は限定的であり、GANによる次の注文の生成は、それなりの精度を達成できることが確認された。また、人工市場シミュレーションのパラメータを変更し、様々な環境での分析を行った結果、市場のノイズが大きい場合には、注文の同時性の効果が消失することが明らかになった。これらの分析を通じて、改めて、前章の人工市場データマイニングプラットフォームの有効性を確認することができた。

第IV部

強化学習を用いたシミュレーションの チューニング

第7章 深層強化学習を用いたマルチエージェントシミュレーションのパラメータチューニング

7.1 研究背景：マルチエージェントシミュレーションにおけるパラメータチューニングの重要性

マルチエージェントシミュレーションにおいて、パラメータチューニングは不可欠な手順である。一般に、パラメータチューニングは以下の2つの目的で行われる。一つ目は、シミュレーションモデルのパラメータを調整することで、実際の現象を再現できるようにすることである。シミュレーションモデルのパラメータには、実世界では直接観測できないものが含まれることが多いため、この手順が必要となる。もう一つは、社会問題の解決に当たって、適切な施策を探索するためのパラメータ探索である。例えば、大規模な渋滞シミュレーションにおいては効率的な人流を作ることを目的としているため、より良い流れを生み出すような様々な施策をパラメータなどのチューニングを通じて探索する。

しかしながら、パラメータチューニングは、パラメータの次元数が多いことが多く、また、必要となるシミュレーション試行も多くなるため、計算コストが大きく、難しいタスクであることが多い。マルチエージェントシミュレーションでは、複数のエージェントを用いてシミュレーションを行うため、各シミュレーション試行の計算コストが比較的多くなることが多い。この点は、いくつかの先行研究でも指摘されている[149, 72]。

しかし、この計算コストの膨大さに対する根本的な解決策は、現時点では提案されていない。例えば、計算コストを低減させるために、Yamashitaら[72]は、マルチエージェントシミュレーションの一部をニューラルネットワークに近似することで、最適解を効率よく獲得する手法を提案している。また、ニューラルネットワークのパラメータチューニングにおいては、ベイズ最適化ベースのTree-structured Parzen Estimator (TPE)[65]などの手法が頻繁に利用される。しかしながら、これらの解決策は、パラメータの次元数を圧縮することのできる探索手法ではないため、次元に対する計算コストの増加の問題を解決できるものではなかった。

また、マルチエージェントシミュレーションでは、エージェントの複雑な相互作用の結果として、カオティックな振る舞いを引き起こす場合が多く、最適パラメータの探索が困難である場合が多い。一方で、マルチエージェントシミュレーションは、複雑な相互作用

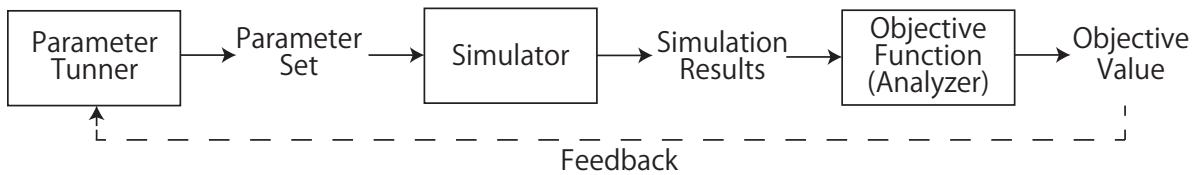
によって生じるカオスな現象を再現することを目的としている場合も多く、大域的最適解に基づいたパラメータ決定が難しいことが多い。

これらの問題を解決するために、本章では、深層強化学習を活用する手法を提案する。近年、高次元の問題を扱うことのできる深層学習手法が多く開発されている。例えば、Bakerら [150] は、オブジェクトを複雑に追加することで、問題を高次元・複雑化したかくれんぼのタスクにおいて、戦略を学習する手法を提案している。このような例を踏まえると、マルチエージェントシミュレーションのような、高次元で複雑な問題に対して、深層強化学習を活用することは、有望な選択肢の一つであると考えられる。

そこで、本章では、マルチエージェントシミュレーションのための強化学習を用いたパラメータチューニング手法を提案する。まずは、第一段階として、本章では、低次元のパラメータのチューニングに着目する。これは、従来の TPE のようなパラメータチューニングを用いた場合でも可能な低次元のパラメータチューニングを行うことで、深層強化学習を用いた手法の性能の比較を行うことができるからである。そこで、本章は、そういった実験を通じて、マルチエージェントシミュレーションのパラメータチューニングに対する深層学習の有効性について示す。

7.2 提案手法

Typical Parameter Tuning Scheme



Proposed Scheme

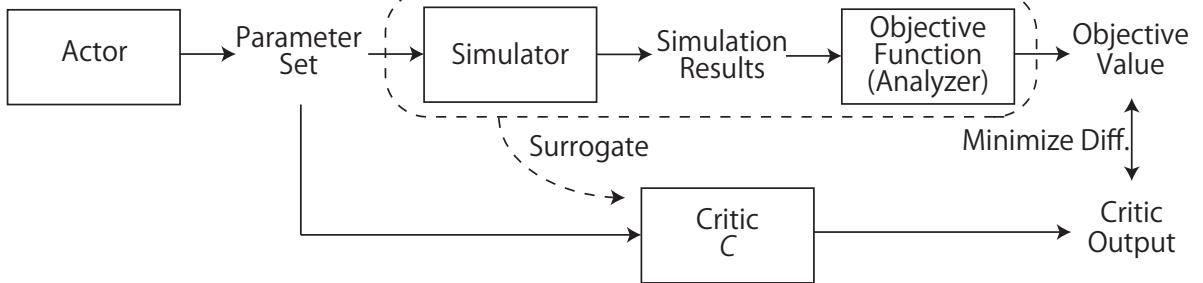


図 7.1: パラメータチューニングのスキーム

提案手法では、連続的な Action 空間をとることのできる Actor-Critic ベースの深層強化学習手法である、Deep Deterministic Policy Gradient (DDPG) [127] と Soft Actor-Critic (SAC) [128, 129] を採用した。これらを採用した理由は以下である。図 7.1 通常のパラメー

タチューニングのスキームと、提案手法のスキームを示した。各試行において、通常のパラメータチューニングのスキームにおいては、チューナーがパラメータセットを生成する。その後、シミュレーターは生成されたパラメータセットを用いてシミュレーションを実行し、結果を得る。その結果を用いて、アナライザーが最終的な目的関数を獲得する。最後に、パラメータチューナーは、その目的関数のフィードバックを受け、目的関数を最大化するようにパラメータセットを修正する。これを深層強化学習に落とし込んで考えると、図7.1で示しているように、Actor-Criticの枠組みがとてもよく合うことがわかる。第二に、一般的なパラメータチューニングのタスクにおけるパラメータセットは、連続値を取ることが多いからである。そのため、連続値を取り扱うことのできる深層強化学習手法であることが必要である。なお、図7.1で示しているものは、手法の概要を示したものであり、モデルによっては正確なものではない。その詳細については、下記で説明する。

DDPGとSACのどちらをそのまま使用しても、マルチエージェントシミュレーションのパラメータチューニングに適合させることはできなかった。そのため、本章では、マルチエージェントシミュレーションのパラメータチューニングに深層強化学習手法を適用できるように、以下の3つの追加要素を提案する。

1. Action Converter (AC)
2. Redundant Full Neural Network Actor (FNNA)
3. Seed Fixer (SF)

Action Converterは、パラメータ空間の上限や下限を固定するために採用するものであり、SACにおいては、Gaussian SamplingのSquashing(後述)により固定できるため、Action ConverterはDDPGに対してのみ適用される。

追加要素の適用パターンについては、表7.1にまとめた。また、表7.2にすべてのパターンのモデルを列挙した。

表7.1: モデルと追加要素の組み合わせパターン

ベースモデル	追加要素		
	AC	FNNA	SF
カスタム DDPG	適用可	適用可	適用可
カスタム SAC	適用せず	適用可	適用可

まず、マルチエージェントシミュレーションのパラメータチューニングのベースとなるモデルである、カスタムDDPGとカスタムSACについて説明し、次に3つの追加要素について説明する。

表 7.2: 全モデル一覧. 一部のモデルは、正常に機能しないため、後述の実験では使用していないものもある.

モデル名	ベースモデル	追加要素		
		AC	FNNA	SF
Baseline	TPE: Optuna	-	-	-
DDPG + AC + FNNA + SF	カスタム DDPG	✓	✓	✓
DDPG + FNNA + SF	カスタム DDPG	-	✓	✓
DDPG + AC + SF	カスタム DDPG	✓	-	✓
DDPG + AC + FNNA	カスタム DDPG	✓	✓	-
DDPG + AC	カスタム DDPG	✓	-	-
DDPG + FNNA	カスタム DDPG	-	✓	-
DDPG + SF	カスタム DDPG	-	-	✓
SAC + FNNA + SF	カスタム SAC	-	✓	✓
SAC + SF	カスタム SAC	-	-	✓
SAC + FNNA	カスタム SAC	-	✓	-
SAC	カスタム SAC	-	-	✓

7.2.1 カスタム DDPG

通常の DDPG と類似しているものの、タスクが異なるため、入出力関係が異なる。そのため、一部機構にも差異が発生する。

まず、このモデルにおける Actor(図 7.2 中の A) は、State を持たない。通常、Actor は現在の State を入力として受け取り、最適な Policy を生成する。しかしながら、本章のパラメータチューニングタスクにおいては、State が存在しない。つまり、パラメータチューニングタスクは、マルコフプロセスなどではないため、ワンショットのトライアルになる。そのため、State が存在しない。加えて、Actor はパラメータセットを生成するが、これは、強化学習でいうところの Action と同等である。これらの変更を踏まえると、提案手法における Actor は、入力を受け付けず、パラメータセットのみを生成するので、以下のように表すことができる。

$$P_i = A() \quad (7.1)$$

ここで、 P_i は i 回目の試行におけるパラメータセットである。パラメータセットが N 次元とすると、 $P_i = (p_{1,i}, p_{2,i}, \dots, p_{N,i})$ である。 A は Actor を示す。

次に、Critic も通常の DDPG と異なる。通常の DDPG の Critic は、State と Action を入力として受け取るが、Actor と同様に、本章のタスクの場合、State を受け取らない。また、通常の Critic の出力目標は Q 値(将来の割引リターンの和)だが、本章のタスクの Critic の

Customized DDPG

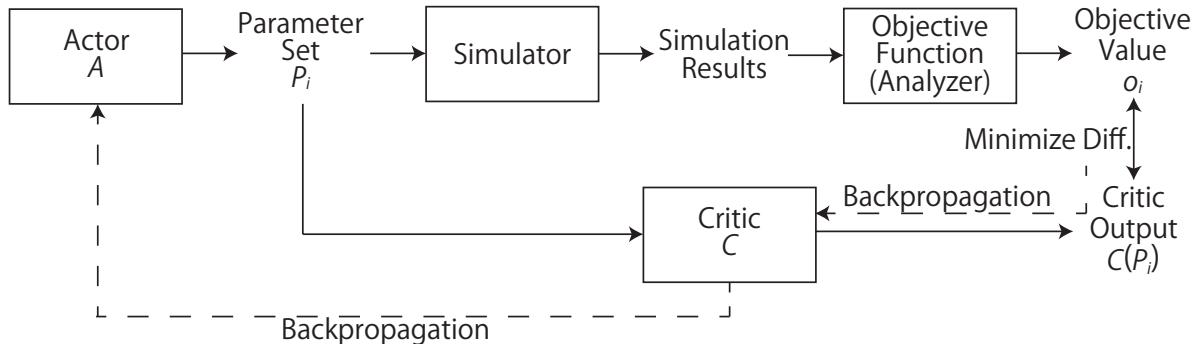


図 7.2: カスタム DDPG のモデル概要

出力目標は、図 7.1 に示した通り、目的関数そのものとなる。これは、パラメータチューニングがマルコフプロセスではなく、ワンショットの試行であるためである。一般的な強化学習のタスクにおいては、Action が半永久的に効果を持つと考え、将来の期待報酬に基づいて各 Action の価値の評価を行う。しかしながら、今回のパラメータチューニングのタスクにおいては、それぞれの試行で、どのようなパラメータを選択しても、サンプリング効率の良し悪しはあるものの、将来の試行に影響は発生しない。そのため、割引後の将来期待報酬を考慮する必要がない。これらを踏まえると、Critic の損失関数は次の通り定義できる。

$$L_C = MSE[o_i, C(P_i)] = (o_i - C(P_i))^2. \quad (7.2)$$

ここで、 o_i は、 i 回目の試行で得られたパラメータセットを用いた場合のシミュレーション結果の目的関数値である。 C は Critic を意味する。 P_i は、 i 回目の Actor によって出力されるパラメータセットである。 $C(P_i)$ は、パラメータセット P_i を Critic に入力した結果の出力値である。

これらに基づくと、Actor の損失関数は以下の通りに定義される。

$$L_A = -C(A()) \quad (7.3)$$

つまり、Actor は、Critic によって推定された目的関数の値を最大化するように学習している。このように、 L_C と L_A の最小化を交互に行うことで、Critic は、パラメータセットを与えられれば、そこに対してより正しい目的関数の値を推定し、Actor は、より高い目的関数の値を得るためにより良いパラメータセットを追求するという形で学習が進む。これらの流れは、オリジナルの DDPG と同じである。

また、オリジナルの DDPG と同様に、探索ノイズとしては Ornstein–Uhlenbeck process [151] を採用し、他にも Replay Buffer [152]、Soft-target Update を採用する。

さらに、シミュレーションが失敗した場合には、目的関数の値を計算できないため、ペナルティーを代わりに付与する。しかし、初期状態のパラメータがあまりにも不適切であ

る場合には、周辺のパラメータを探索しても同様にシミュレーションが失敗するため、勾配の獲得が困難になる。そこで、勾配が取得できないような状況に陥った場合には、パラメータをランダムに初期化しなおすこととする。

7.2.2 カスタム SAC

オリジナルの SAC [128, 129] をベースとして、本章のタスクに合わせた、カスタム SAC を作成した。カスタム DDPG と同様に、入出力が変更されていることが主要な変更点である。しかしながら、DDPG と異なり、SAC は DDPG よりも若干複雑なアーキテクチャを持つ。

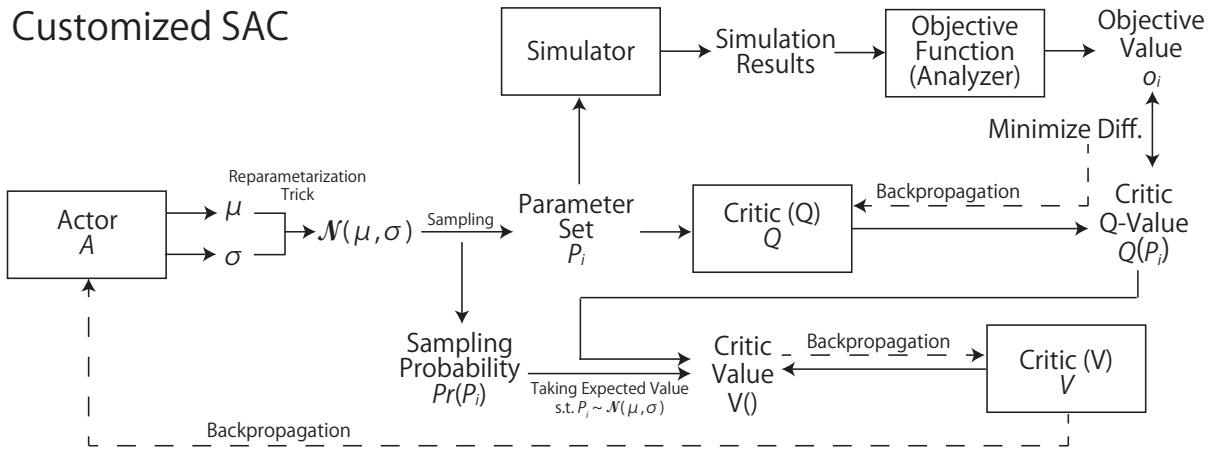


図 7.3: カスタム SAC のモデル概要

SAC は、連続的な Action 空間に対応しているという点では、DDPG と同様である。しかしながら、SAC は、DDPG と異なり、確率的なポリシーを採用しているほか、目的関数に、高い探索能力を実現する、エントロピー項を付加しているという点で異なる。

まず、Actor は、再パラメータ化トリックを用いて、以下の通り、パラメータセットを生成する。

$$\mu, \sigma = A(), \quad (7.4)$$

$$P_i \sim \mathcal{N}(\mu, \sigma). \quad (7.5)$$

ここで、 $A()$ は、SAC の Actor であり、 $\mathbb{R}^{2 \times N}$ の出力を持つ。 $\mathcal{N}(\mu, \sigma)$ は、ガウス分布であり、平均と分散が $\mu \in \mathbb{R}^N$ と $\sigma^2 \in \mathbb{R}^N$ で定義される。さらに、 $P_i = (p_{1,i}, p_{2,i}, \dots, p_{N,i})$ は、 i 回目の試行の N 次元のシミュレーションパラメータセットである。

SAC の Critic は、カスタム DDPG とはことなり、 $Q(P_i)$ と $V()$ の二つのネットワークを持つ。 $V()$ は入力を受け取らず、その目標値は下記の通り定義される。

$$V() = \mathbb{E}_{P_i \sim \mathcal{N}(\mu, \sigma)} [Q(P_i) - \alpha \ln \{Pr(P_i)\}]. \quad (7.6)$$

ここで, $\Pr(P_i)$ は, 式 7.5において, パラメータセット P_i が $\mathcal{N}(\mu, \sigma)$ からサンプリングされる確率を示しており, α はハイパーパラメータである. 本章の実験では, $\alpha = 0.2$ に設定した. 通常の強化学習手法においては, $Q(P_i)$ と $V()$ は, 行動価値関数と状態価値観数として解釈できるが, 本章のタスクにおいては, State が存在しないため, そのように解釈することはできない. あえて, それらを解釈をするのであれば, $Q(P_i)$ と $V()$ は, パラメータ価値関数と探索価値関数と解釈し得る. これは, $V()$ の第二項が, Policy Entropy 項であり, 探索に対するエントロピーを評価する項であるからである. また, P_i は, 式 7.6 の右辺に表れているが, $V()$ は P_i を引数に取らない関数である. これは, $V()$ が, 一般的な強化学習の理論においては, 現在の状態価値を評価するものであり, Action を引数に取らないためである. 本研究においては, State が存在しないために, 一切の引数を持たない関数として表現されている.

Critic の損失関数は以下の通り, 定義される.

$$L_Q = MSE[o_i, Q(P_i)] = (o_i - Q(P_i))^2, \quad (7.7)$$

$$L_V = MSE[Q(P_i) - \alpha \ln \{\Pr(P_i)\}, V()] \quad (7.8)$$

$$= [Q(P_i) - \alpha \ln \{\Pr(P_i)\} - V()]^2. \quad (7.9)$$

本章の実験においては, ターゲットネットワークを採用した. そのため, 式 7.8 と式 7.9 における $Q(P_i)$ は, 固定されたネットワークであり, 勾配をとらない.

Actor の損失関数は以下となる.

$$L_A = -V() = -\mathbb{E}_{P_i \sim \mathcal{N}(A())} [Q(P_i) - \alpha \ln \{\Pr(P_i)\}]. \quad (7.10)$$

ここで, 誤差逆伝播は, 前出の再パラメータ化トリックにより実現される.

オリジナルの SAC[128, 129] においては, Action 空間の Squashing, つまり, Action 空間に制約をかける手法が採用されていた. 本章の実験においても, パラメータに正値の制約をかける. そのため, squashed Gaussian policy を定式化する. 本章の実験においては, $f(x) = \ln(1 + \exp(x))$ という関数を採用する. そのため, Gaussian Policy は下記の通り変換される.

$$\ln \{\Pr(P_i)\} = \ln \{\Pr_{\mathcal{N}}(\mathbf{u})\} + \sum_j P_{i,j} - \sum_j u_j. \quad (7.11)$$

ここで, $P_i = \ln(1 + \exp(\mathbf{u}))$, $\mathbf{u} \sim \mathcal{N}(\mu, \sigma)$ であり, $\Pr_{\mathcal{N}}(\mathbf{u})$ は, サンプリングの確率である. この定義は, Action Squashing を採用しているため, 式 7.5 の定義とは異なる.

加えて, カスタム SAC においては, Replay Buffer [152], Soft Target および, シミュレーションの失敗時のペナルティーが, カスタム DDPG と同様に適用する.

7.2.3 Action Converter (AC)

カスタム DDPG に対して, パラメータ空間の制約をかけるために, Action Converter を提案する. Action Converter は, SAC における Action Squashing と同様に, Actor の

出力を制限するためのものである。本章の実験では、正のパラメータ制約に対して、カスタム SAC と同様に、 $f(x) = \ln(1 + \exp(x))$ という関数を用いて制約を実現する。

これは、前出の SAC における Enforcing Action Bounds [128, 129] と同様である。しかしながら、Action 空間の制約という目的だけは、AC と共にしているものの、SAC は決定論的な強化学習ではなく、DDPG は決定論的な強化学習という点で大きな違いがある。その結果、SAC では Action 確率を変換一方で、DDPG は行動確率を持たないので、Action 空間をそのまま変換することとなる。

そこで、DDPG に対する Action 空間の制約は、SAC よりも緩和した形で実装することとした。これは、特に制約を必要としないパラメータ領域においては、DDPG の探索を阻害しないよう、過度な行動空間の変更を避けるためである。これは、正の制約において $f(x) = \ln(1 + \exp(x))$ を採用したことにつながっている。これは、 $\frac{\partial f(x)}{\partial x} = \frac{e^x}{1+e^x} = 1 - \frac{1}{1+e^x}$ であるため、 x が十分に大きい場合には、勾配に影響が出にくからである。

もちろん、ここで、別の関数を採用することは可能であるが、ここでは、他の関数については議論を行わないこととする。

AC の実装にあたっては、Actor の出力に対して、パラメータ制約を直接適用することで実装する。たとえば、パラメータ空間が 2 次元であり、正の制約を与えることとした場合を考える。この場合、 $P_i = (p_{1,i}, p_{2,i})$ としたときに、 $p_{1,i}^* = \ln(1 + \exp(p_{1,i}))$ と $p_{2,i}^* = \ln(1 + \exp(p_{2,i}))$ のように変換を行うこととなる。そして、最終的なパラメータセットは、 $P_i^* = (p_{1,i}^*, p_{2,i}^*)$ となり、 P_i の代わりに使用されることとなる。

ただし、AC はカスタム DDPG にのみあることに注意されたい。

7.2.4 Redundant Full Neural Network Actor (FNNA)

提案手法のモデルにおける Actor の最も単純な実装は、勾配計算が有効になっているパラメータを返すだけである。これは、カスタム DDPG でもカスタム SAC でも同様であり、図 7.4 の左の図の通りである。つまり、提案手法における Actor は入力を取らないため、パラメータの次元数が N である場合の最小の Actor の実装は、勾配付きの N 個のパラメータそのものとなる。

しかしながら、本章では、Redundant Full NN Actor (FNNA) という、冗長なニューラルネットワークを用いた Actor を提案する。図 7.4 の右の図が示している通り、FNNA は、MLP を内包しており、入力として、すべて 1 で満たされた、ダミーベクトルを採用したネットワークを採用する。

一見すると、このネットワークは、冗長で無駄であるように見えるが、これは宝くじ理論 [153] にも似ているとも考えることができ、深層学習の学習を通じて、複数の冗長なリンク構造の中から、集合知のように有用な出力を生成するとも考えることができる。これにより、Actor の学習の安定性が実現できるのではないかと考えた。

加えて、カスタム SAC をベースモデルとして採用する場合には、この冗長な FNN は、V-net ($V()$) にも適用される。

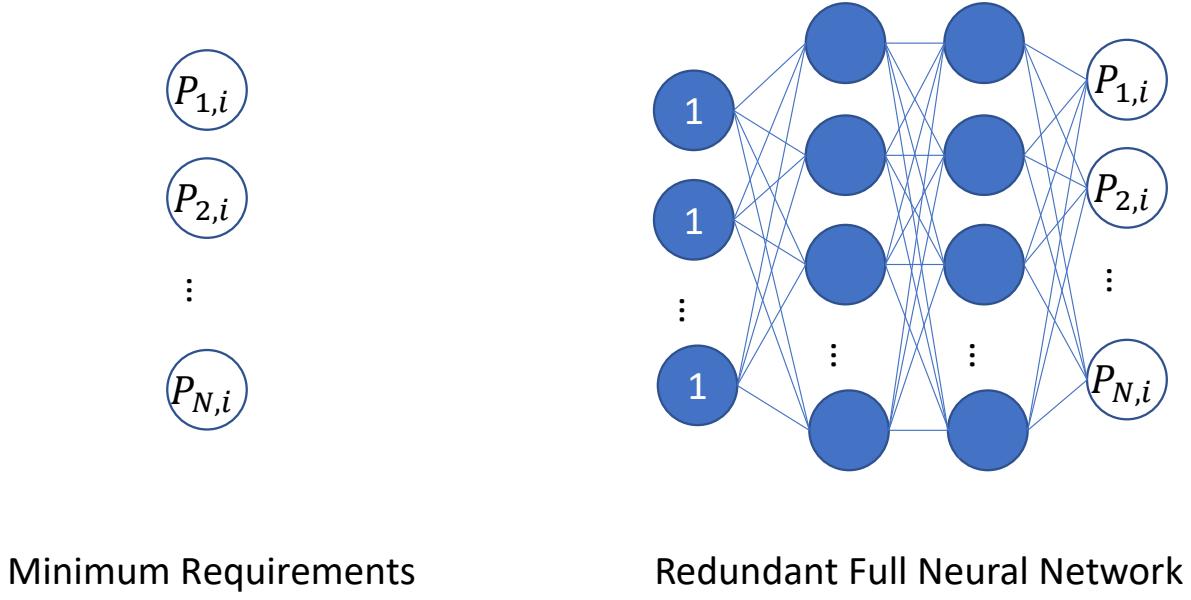


図 7.4: FNNA のイメージ

7.2.5 Seed Fixer (SF)

マルチエージェントシミュレーションは、シミュレーションのランダムシードに依存して、様々な挙動を示すことが多い。多くのマルチエージェントシミュレーションにおいては、エージェントの確率的な意思決定を実現するために、乱数が行動アルゴリズムに使用されている。これらの行動決定におけるわずかな差異が、エージェント間の複雑な相互作用を通じて、カオス的に大きな差異を引き起こし得る。そのため、ランダムシードの違いは、図 7.5 の左図のように、大きな違いを引き起こす可能性を秘めている。

特に、深層学習において、ランダムシードの影響は大きい。そもそも Critic は、シミュレーションのパラメータセットとシミュレーションの結果の関係を学習するために導入されている。そして、Actor は Critic を通じて獲得できる勾配を用いて学習を行う。このとき、Critic の勾配がなだらかであること、つまり、入力パラメータに対する Critic の出力の応答がなめらかであることが不可欠である。そのため、パラメータセットの小さな差による出力の変化を正しく学習できるように、入力に対する出力の応答は、ランダムシードの影響よりも十分に大きい必要がある。

一般に、シミュレーションの回数を増やすと、ランダムシードの影響は緩和されるが、Critic の勾配に対しては、相当数試行回数を浸す必要が出てくる。また、シミュレーションの一回当たりの計算コストが大きい場合には、シミュレーション回数を増やすことは困難である。

Seed Fixer は、図 7.5 の右図のように、各シミュレーションで使用するランダムシードを固定することで、これらの問題を解決する。図 7.1 中のシミュレーションを通じて目的

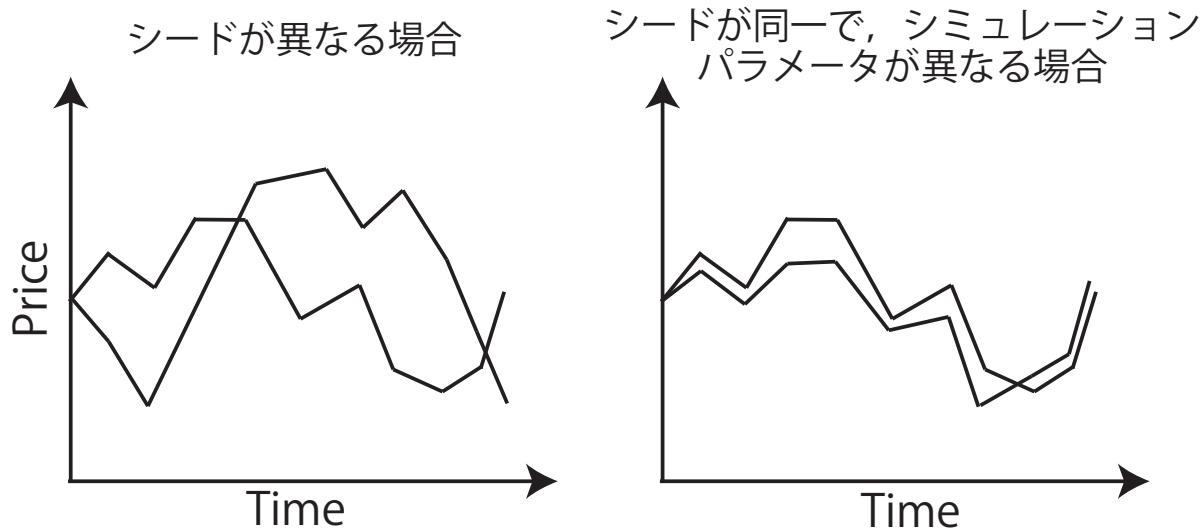


図 7.5: Seed Fixer のイメージ. この例では, 人工市場シミュレーションの各価格系列をプロットしている. 左図では, シミュレーションのランダムシードが異なり場合のイメージで, 右図はシミュレーションのランダムシードが同じであるが, シミュレーションのパラメータのみが少し異なる場合のイメージである.

値を得る過程では, M 回の試行の平均で計算を行う. このとき, Seed Fixer を採用した場合には, この M 個のシミュレーションのランダムシードの組を固定する.

Seed Fixer を導入することで, ランダムシードの影響がなくなり, Critic の勾配を滑らかにすることができます. マルチエージェントシミュレーションにおいては, パラメータのほんの少しの差だけでもカオスなふるまいを示すこともあるため, 完全な解決にはならないものの, 勾配の滑らかさという観点では, 大幅な改善を図ることができる.

しかしながら, この Seed Fixer は, 学習結果にバイアスが発生するというトレードオフが存在する. 学習において, ランダムシードは M パターンに固定され, この M このパターンでは, 起こり得る事象をカバーするには少なすぎる結果, 偏りが発生する可能性がある. そのため, Seed Fixer は, この偏りの発生可能性と, Critic の勾配の滑らかさのトレードオフの関係にある. より M を大きくすることは, 1つの解決策になり得るが, 前述の通り, 計算コストの問題が発生する可能性もあり, 得策ではない. しかしながら, 本章の実験では, $M = 100$ でも, 大きな偏りが発生することがなかったため, この点については, ひとまず問題ないものとみなした.

7.3 実験

7.3.1 タスク設定

実験には、マルチエージェントシミュレーションとして、人工市場シミュレーションを採用した。

シミュレーションモデル

5.3.1 節で用いたものと同様の人工市場モデルを使用する。なお、 $w_F^i \sim Ex(w_F)$ 中の w_F と $w_C^i \sim Ex(w_C)$ 中の w_C を実験でチューニングのターゲットとする。

目的関数

本実験において、目的関数は、対数リターンの尖度と歪度の誤差とすることとした。また、歪度 (Skewness) と尖度 (Kurtosis) は以下のように計算される。

$$x_t := \ln \{p_t/p_{t-1}\}, \quad (7.12)$$

$$\text{skewness} = \frac{1}{T} \sum_{t=1}^T \frac{(x_t - \bar{x})^3}{(x_t - \bar{x})^{2\frac{3}{2}}}, \quad (7.13)$$

$$\text{kurtosis} = \frac{1}{T} \sum_{t=1}^T \frac{(x_t - \bar{x})^4}{(x_t - \bar{x})^{2^2}}. \quad (7.14)$$

ここで、 T は、シミュレーションステップ数であり、本実験では、10,000 とした。

先行研究の分析 [154] に基づいて、歪度と尖度のターゲットをそれぞれ 0.0 と 6.0 に設定した。これらの数値は、それぞれの市場によって異なるため、大まかに設定をした。本研究では、提案手法によるチューニングの性能を分析したいため、この設定の厳密性については、必ずしも重要ではない。

最終的な目的関数は、歪度と尖度の目標値との差の Mean Square Error (MSE) とした。そのため、次のように計算される。

$$\text{MSE}[(\text{Skewness}, \text{Kurtosis}), (0.0, 6.0)] = (\text{Skewness} - 0.0)^2 + (\text{Kurtosis} - 6.0)^2. \quad (7.15)$$

もし、シミュレーションが途中で失敗した場合には、この MSE は、ペナルティーに置き換える。本実験では、このペナルティーを 1,000 に設定した。なお、シミュレーションの失敗は、パラメータ設定の不適切さゆえに、価格が無限大に発散するなどの原因で発生する。

最終的な目的関数は、負の MSE であり、最大化する方向でチューニングを行う。

チューニングにおいては、すべてのシミュレーションを 100 回実行し、その平均値を最終的な出力として使用することで、目的関数値の平滑化を行う。また、チューニング後の最終評価に当たっては、1,000 回のシミュレーションを実施して、性能を評価する。

7.3.2 モデル

前述の通り、本実験におけるパラメータチューニングのタスクは、目的関数を最大化する、よりよい w_F, w_C を探索することである。図 7.1 に示している通り、ベースラインの一般的なチューニングモデルとして、ベイズ推定ベースの tree-structured Parzen estimator (Optuna) を採用した。以下では、その詳細と、提案手法の詳細な実装と設定について説明する。

ベースライン: Tree-structured Parzen Estimator (Optuna)

マルチエージェントシミュレーションのパラメータチューニングのベースラインモデルとして、Tree-structured Parzen Estimator (TPE) [65] を採用した。このモデルは、機械学習のパラメータチューニングによく使われるもので、Optuna [66] として知られている。TPE は、ブラックボックス最適化問題において、より良い目的関数値を探査するためのベイズ最適化手法の一種である。

カスタム DDPG

学習離村等については、すでに、7.2 節で述べた通りである。ここでは、実験における、カスタム DDPG のモデルパラメータなどの実装の詳細について説明する。

- Actor (FNNA ありの場合): 4 層の MLP でからなる構造をもつ。隠れ層は 100 次元であり、出力は 2 次元である。最終層を除くすべての層では、Layer Normalization と ReLU を用いる。FNNA を用いるため、入力としては、100 次元のすべての要素が 1 のダミーベクトルを採用する。
- Actor (FNNA なしの場合): 勾配付きパラメータを返す
- Critic: 100 次元の隠れ層をもつ、4 層の MLP である。入力としては、2 次元のパラメータセットを受け取り、目的関数値の推定値を 1 次元で出力する。最終層を除くすべての層では、Layer Normalization と ReLU を用いる。
- DDPG の Soft Target Update Ratio: 0.1
- Actor の学習率: 10^{-4}
- Critic の学習率: 10^{-3}
- Batch Size: 100
- Experience Replay のバッファーサイズ: 1,000

カスタム SAC

学習理論等については、すでに、7.2節で述べた通りである。ここでは、実験における、カスタム SAC のモデルパラメータなどの実装の詳細について説明する。

- Actor (FNNA あり): 4層の MLP でからなる構造をもつ。隠れ層は 100 次元であり、出力は μ と σ に対応する 2 次元の出力 2 個である。最終層を除くすべての層では、Layer Normalization と ReLU を用いる。FNNA を用いるため、入力としては、100 次元のすべての要素が 1 のダミーベクトルを採用する。
- Actor (FNNA なし): μ と σ に対応する勾配付きパラメータを返す
- Q-net ($V(P_i)$): 100 次元の隠れ層をもつ、4 層の MLP である。入力としては、2 次元のパラメータセットを受け取り、目的関数値の推定値を 1 次元で出力する。最終層を除くすべての層では、Layer Normalization と ReLU を用いる。
- V-net ($V()$. FNNA ありの場合): 4 層の MLP でからなる構造をもつ。隠れ層は 100 次元であり、出力は 1 次元 ($V()$) である。最終層を除くすべての層では、Layer Normalization と ReLU を用いる。FNNA を用いるため、入力としては、100 次元のすべての要素が 1 のダミーベクトルを採用する。
- V-net ($V()$. FNNA なしの場合): $V()$ に対応する 1 次元の勾配付きパラメータを返す。
- SAC の Soft Target Update Ratio: 0.1
- Actor の学習率: 10^{-4}
- Critic の学習率: 10^{-3}
- Batch Size: 100
- Experience Replay のバッファーサイズ: 1,000

7.3.3 評価

評価に当たって、どのモデルでも、学習に利用できるシミュレーション数は、10 万回に制限した。これは、1 回の学習で、100 回のシミュレーションを要する場合、1,000 回の学習が行われることになる。

学習終了後、学習時に最も良かったモデルを用いて、1,000 回のシミュレーションを行い、最終的な評価を行う。

7.4 結果

表 7.3: 評価結果 (10 試行の統計値)

Model	Loss ($-o_i$)			Kurtosis		Skewness	
	Mean	Median	Mean	Median	Mean	Median	
Baseline (TPE: Optuna)	3.410830 ± 5.993605	0.558568	6.629670 ± 0.437744	6.407023	0.006460 ± 0.009737	0.006179	
DDPG + AC + FNNNA + SF	0.727502 ± 0.560671	0.489755	6.288674 ± 0.092182	6.158437	0.007488 ± 0.004775	0.004005	
DDPG + FNNNA + SF	Cannot be tuned at all						
DDPG + AC + SF	17911.68 ± 21985.85	4942.005	26.35092 ± 23.80422	7.160722	0.014936 ± 0.037776	0.000424	
DDPG + AC + FNNNA	1094.991 ± 3252.000	0.652749	8.631269 ± 4.523929	6.892526	0.005324 ± 0.020119	0.010744	
DDPG + AC	36871.57 ± 47942.91	20397.35	44.59121 ± 45.65828	7.138884	-0.035963 ± 0.081353	-0.001542	
SAC + FNNNA + SF	0.194473 ± 0.045963	0.195069	6.245487 ± 0.087551	6.222321	0.006581 ± 0.004671	0.004943	
SAC + SF	0.203258 ± 0.045476	0.209067	6.275684 ± 0.050873	6.243806	0.007037 ± 0.004622	0.006978	
SAC + FNNNA	21.41606 ± 14.70367	25.60599	9.855362 ± 1.985068	9.583112	0.009523 ± 0.008616	0.007500	
SAC	17.45062 ± 14.48998	21.85446	9.363488 ± 2.028399	9.176651	0.007901 ± 0.006655	0.008429	

表7.3は、10試行の結果を示している。この表は、すべてのモデルのチューニング結果の損失(負の目的関数値)、歪度、尖度を示している。また、平均値と標準偏差、中央値を代表値として使用している。

損失は、式7.15で定義され、損失が小さいほど良いパラメータになっていることを示す。また、前述の通り、Actor-Criticのフレームワークにおいて、通常は、目的関数値を最大化するため、損失は目的関数の正負を入れ替えたものとなる。今回の設定では、この損失関数にMSEを使用しているため、損失は非負の値となる。

表7.3の結果によれば、SAC + FNNA + SFが最も良い結果を示している。SAC + FNNA + SFは、損失の平均だけでなく、損失の標準偏差や中央値の観点でも、ベースラインや他のモデルよりも良い結果となっている。

全体を通してみると、SACベースのモデルが全体的に良い結果を示したが、DDPGベースのモデルでもベースラインを上回ることはできた。しかしながら、SACベースのモデルは、DDPGベースのモデルよりも安定している結果となった。

DDPGから追加要素を一部取り外したモデルの結果を考慮すると、本章で提案した、AC, FNNA, SFのすべての要素がDDPGベースのモデルには不可欠であることがわかる。それらのうち、1つでも要素が欠けると、性能が大きく劣化することが確認できた。特に、ACが欠けると、学習が全く行われないため、ACがチューニングには不可欠であることが分かった。これは、ACが欠けた場合に、タスクにおける、パラメータの非負制約が満たされない場合が発生するために、シミュレーションが成立しないことによるものであると考えられる。その結果として、常に固定値のペナルティーが目的関数値として出力されるようになり、勾配の方向を取れなくなってしまう。そのため、少なくとも、今回のタスクにおいて、ACはDDPGベースのモデルに必要不可欠な要素であると言える。

それぞれの追加要素の効果の比較をすると、DDPGベースのモデルの場合、AC > FNNA > SFの順で、効果が大きいと言える。SFの効果は、比較的小さいように見えるが、SFが欠落している場合には、DDPGベースモデルではベースラインを上回ることができない。

一方、SACベースのモデルでは、SFが必要不可欠な追加要素である。FNNAは、SACベースのモデルがSFを採用している場合にのみパフォーマンスの向上が確認できた。DDPGベースのモデルと異なり、今回は、SACでは、Squashed Gaussian Policyを採用したため、ACは適用しなかった。しかし、ACを除いた場合の追加要素の影響の大きさは、DDPGベースのモデルと同じ結果となった。

図7.6から7.14は、すべてのモデルの各epochでの損失関数値の推移を示している。ここでは、プロット作成用に、特別に学習とは別に確保した評価用の1,000シミュレーションを用いてepochごとのモデルの損失関数値をプロットした。図7.6は、ベースライン(TPE: Optuna)の結果を示している。この図を見ると、学習中に、学習サンプルにオーバーフィットしたような現象が確認できる。一方、図7.7では、DDPGベースのモデル(DDPG + AC + FNNA + SF)の収束の遅さを観測できる。これらのモデルとは異なり、図7.11に示しているSACベースのモデル(SAC + FNNA + SF)では、より早く、安定した収束が進んでいることが確認できる。また、これらの図を比較すると、SACベース

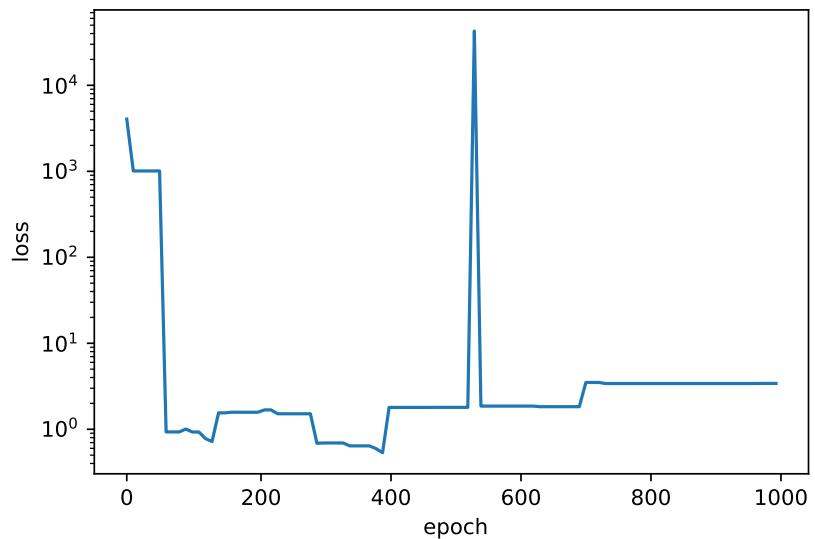


図 7.6: ベースラインモデル (TPE: Optuna) の損失関数の推移

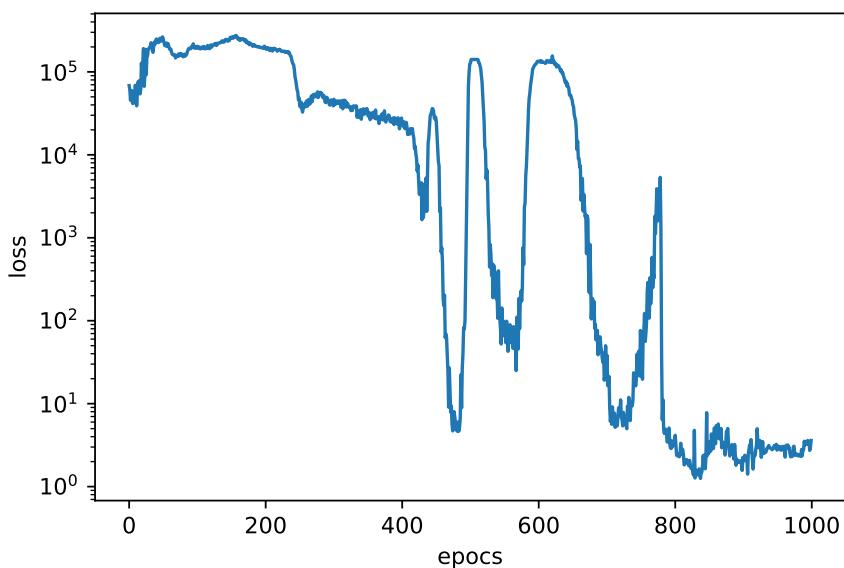


図 7.7: DDPG + AC + FNNA + SF の損失関数の推移

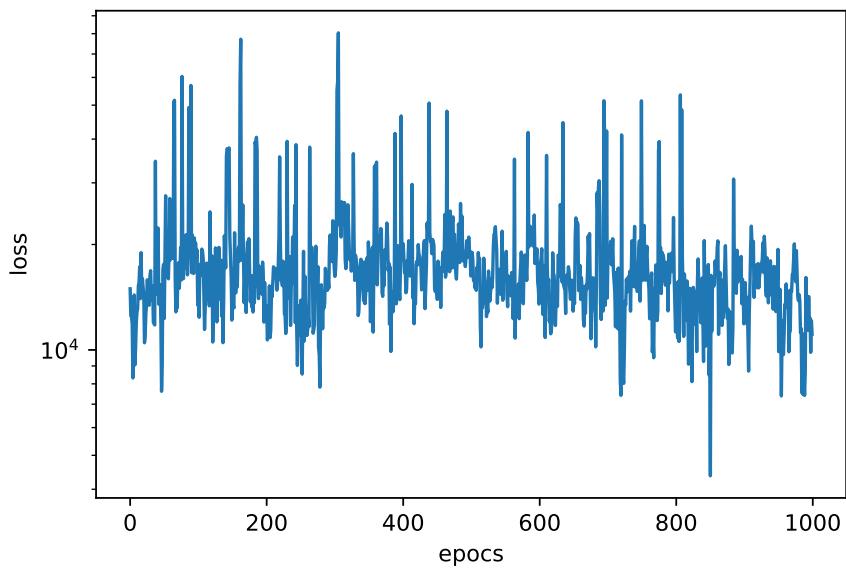


図 7.8: DDPG + AC + SF の損失関数の推移

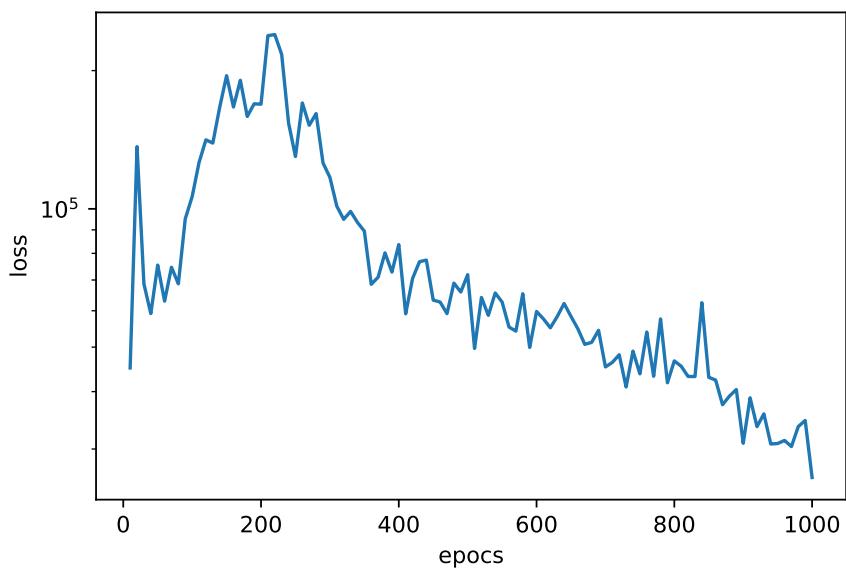


図 7.9: DDPG + AC + FNNA の損失関数の推移

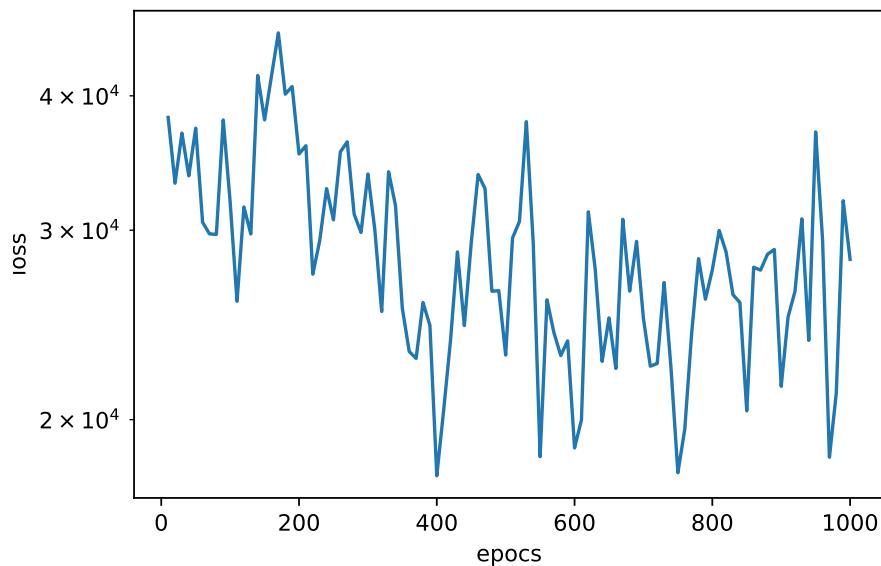


図 7.10: DDPG + AC の損失関数の推移

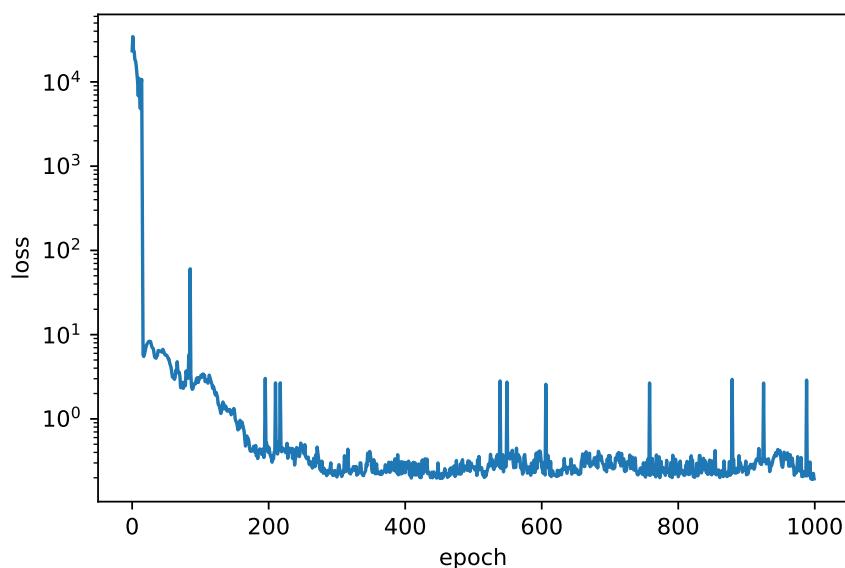


図 7.11: SAC + FNNA + SF の損失関数の推移

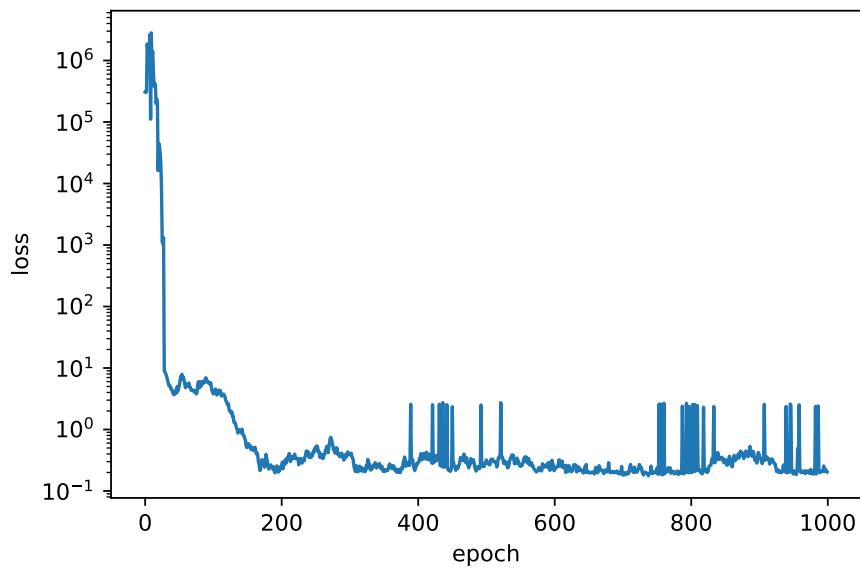


図 7.12: SAC + SF の損失関数の推移

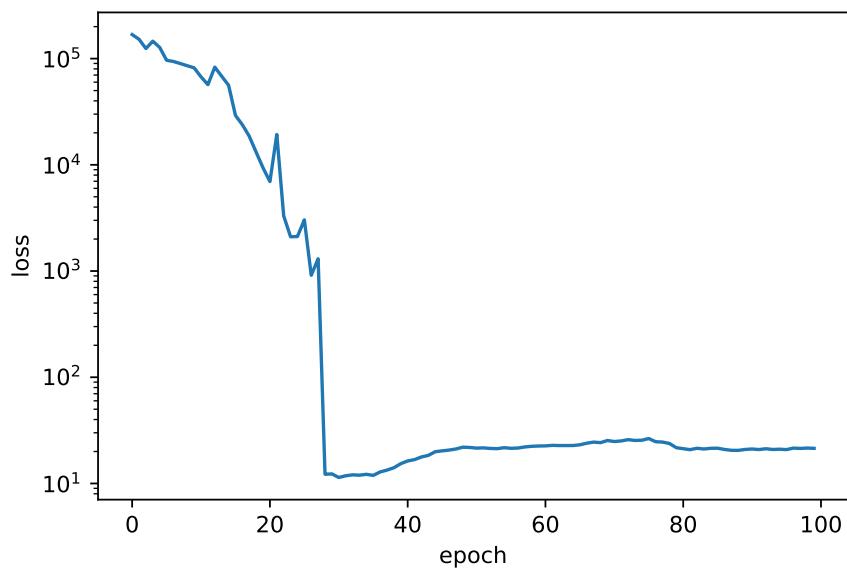


図 7.13: SAC + FNNA の損失関数の推移

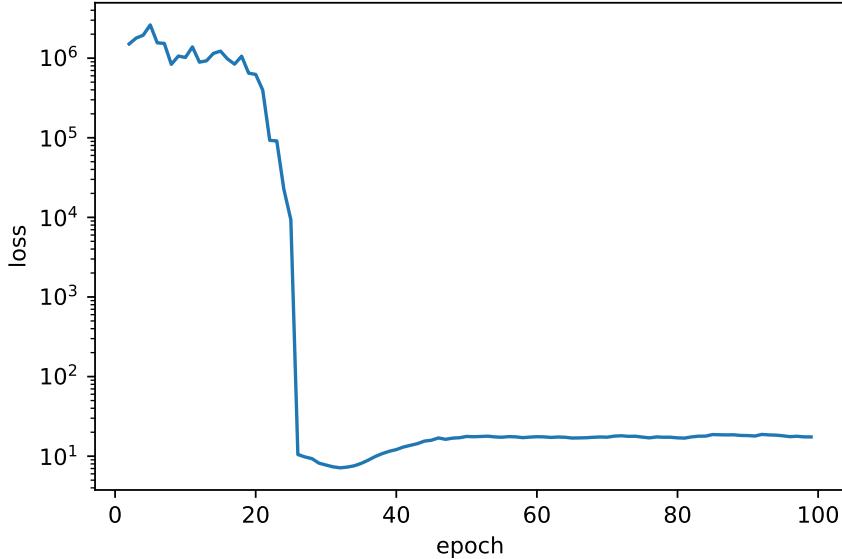


図 7.14: SAC の損失関数の推移

のモデルは、他のモデルよりも性能と収束速度の観点で優れていると言える。また、図 7.11–7.14 を比較すると、Seed Fixer がより速い収束に寄与していることもわかる。

7.5 考察

まず、提案手法は、ベースラインモデルよりも良いチューニング性能を示した。本章での評価では、SAC ベースのモデルは、特に Seed Fixer を用いた場合に、最高の性能を示した。本章でのタスク設定は、次元数が少ないため、比較的、ベースライン手法の方が有利である可能性はある。しかしながら、提案手法が適切に機能している結果を得られたことができたので、提案手法の可能性が示された。

さらに、DDPG ベースのモデルには 3 つの追加要素が、SAC には Seed Fixer が必須であることが分かったが、これらに共通することとして、Seed Fixer は必要不可欠であるということが言える。SAC ベースのモデルでさえも、Seed Fixer がなければ、ベースラインモデルを上回る性能は達成できていない。また、図 7.11 と図 7.13、図 7.12 と図 7.14 を比較すると、Seed Fixer による収束の高速化が明確に確認できる。これらの結果から、マルチエージェントシミュレーションのパラメータチューニングにおける、Seed Fixer の重要性が分かった。本章では、マルチエージェントシミュレーションが複雑系として、カオスなふるまいをする可能性を考慮し、ランダムな確率変数の違いが大きな違いを生む可能性を仮定し、乱数シードを固定した勾配獲得手法を提案した。今回の結果から、この仮定と提案手法の有効性が示されたと言える。

また、結果を総合すると、SACベースのモデルは、DDPGベースのモデルよりも優れていると言える。第一に考えられる理由は、SACが決定論的手法ではないことである。DDPGは、決定論的な手法であるため、DDPGベースのモデルは、連続的な探索のみによってパラメータチューニングを実現しているが、今回のタスクにおけるパラメータ空間は非常に広いため、連続的な探索によって、最適解を見つけることは非常に困難である。しかし、SACでは、確率的な探索が可能であるために、結果に大きな差が発生したと考えられる。加えて、第二の理由は方策エントロピーにあると考えられる。SACは方策エントロピー項(式7.6の第2項)を持つため、強力な探索能力を維持できる。したがって、DDPGベースのモデルと比較して、SACベースのモデルは、かその高い探索能力ゆえに、本タスクにおいて、より良い結果を示したと考えられる。

本章の実験を通じて、Actor-Criticベースの手法がマルチエージェントシミュレーションのパラメータチューニングに使用可能であることを示したが、このことは、Criticが、シミュレーションとアナライザの近似関数として機能していると考えることができる。図7.1で示した通り、提案手法におけるActorは、シミュレーションの出力から直接フィードバックを受けて学習することはない。一方で、Actorの学習に当たっては、Actorの出力に対するCriticの評価値を通じてフィードバックを受ける。このことは、Criticがシミュレーションの近似として十分に機能しており、そのおかげでActorの学習において、勾配情報を引き継げないシミュレーションを通じて学習を行うことを回避できている。これは、CriticがEnd-to-endのサロゲートモデルとしてうまく機能していることを意味しており、このCriticのサロゲートにより、Actorがパラメータチューニングの勾配を獲得できたと言える。

評価の観点でいうと、本章での評価は、それぞれのモデルを評価するにはまだ不十分であり、まだ改良の余地がある。本章での実験では、単なる尖度と歪度のチューニングでしかなかったが、このチューニングは実用的ではない。そのため、これをより実用的なチューニングタスクに変えて実験を行うことが望ましいと考える。これについては、次章で取り組む。

さらに、本章での結果を踏まえると、より高次元のパラメータチューニングのタスクでモデルの検証を行う必要が考えられる。深層強化学習の文脈でいえば、先行研究で、様々に高次元のタスクで、深層強化学習が高い性能を発揮してきている。そのため、提案手法も、より高次元のタスクでも、高い性能を発揮する可能性が高い。一方で、サロゲートモデルの文脈で考えると、パラメータ次元数が大きい場合には、Criticによるシミュレーションの近似がより複雑になるために、近似関数の学習のために必要なデータ数が増える可能性がある。そのため、これらを総合すると、本章での提案手法が高次元のタスクに対して有効かどうかについては、未知な点が多く、さらなる検証が必要であると考えられる。そこで、次章では、より高次元で現実的なタスクについて取り組む。

7.6 本章のまとめ

本章では、カスタマイズされた強化学習を用いて、マルチエージェントシミュレーションのパラメータチューニング手法を提案した。提案手法では、DDPGやSACなどのActor-Criticベースの強化学習手法をマルチエージェントシミュレーションのパラメータチューニングようにカスタマイズした。さらに、AC, FNNA, SFとよぶ、3つの学習を安定化させる追加要素を提案した。実験では、人工市場シミュレーションを用いた。チューニングの目的関数は、尖度と歪度が現実的な値に近くなるように、その差異の負のMSEを採用した。実験では、提案手法と、ベイズ推定ベースのTPE(Optuna)をベースラインとして、比較した。その結果、提案手法がベースラインを上回ることが確認できた。特に、SACを用いたモデルは、ベースラインを含む他のモデルよりも優れていることが確認できた。これにより、提案手法の有効性が示せた。さらに、DDPGベースのモデルでは、AC, FNNA, SFが、SACベースのモデルではSFが必要不可欠な追加要素であることが示された。さらに、Criticがシミュレーションのサロゲートモデルとしてうまく機能していると考えられ、そのおかげで、Actorが適切なパラメータを学習できると考えられる。今後の課題として、学習のさらなる安定性や、高次元パラメータのタスクへの適用などの課題がある。次章ではこれらのうち、より現実的な高次元のチューニングタスクに取り組む。

第8章 GANによる評価と強化学習の融合

8.1 本章のモチベーションと目的

本章では、4章で示した、GANを用いたシミュレーションの妥当性評価機構である、GAN Evaluatorと、7章で示した、強化学習機構を用いた、シミュレーションのパラメータチューニング手法の融合を行う。

従来のマルチエージェントシミュレーションにおいては、シミュレーションの妥当性を担保するパラメータのチューニングというのが困難であった。これは、一般に、逆シミュレーション問題とも言われており、現実と同じ現象を再現できるようなパラメータを見つける問題であり、シミュレーションがパラメータから現象への写像と考えた場合に、現実的な現象に対するシミュレーションの逆関数を見つける取り組みともいえる。

このシミュレーションのパラメータチューニングにおける問題は、現象の定量化が難しいことと、パラメータチューニング自体の難しさが混在していた。しかしながら、4章で示した、GANを用いたシミュレーションの妥当性評価機構では、前者の現象の定量化を実質的に達成したともいえる。さらに、7章で示した、強化学習機構を用いた、シミュレーションのパラメータチューニング手法は、後者のチューニングの難しさを解決する手法ともいえる。そのため、これらの技術を融合させることにより、現状のマルチエージェントのシミュレーションの難しさを解決できる可能性があると言える。

特に、7章における、パラメータチューニングの強化学習的アプローチにおいて、Critic自身が、シミュレーションのサロゲートになっていることを考慮に入れると、このサロゲート関数の逆関数が逆シミュレーション問題で明らかにしたい関数であり、パラメータをActorに出力させることには、整合性の取れるアプローチであるともいえる。

本章では、融合に当たって、4章のGAN Evaluatorを通じて、チューニングの目的関数値を獲得することで、現実的なシミュレーションを達成するパラメータ探索を行う。

ただし、本章では、4章の結果に基づき、GAN Evaluator値の出力値をシミュレーションの妥当性を評価する定量指標としてみなすこととする。つまり、本章では、改めて、Stylized Factsなどの再現の確認は行わず、GAN Evaluatorの出力値の最大化問題を通じ、現状の技術のパフォーマンスおよび限界について議論を行うこととする。

8.2 手法

Proposed Scheme

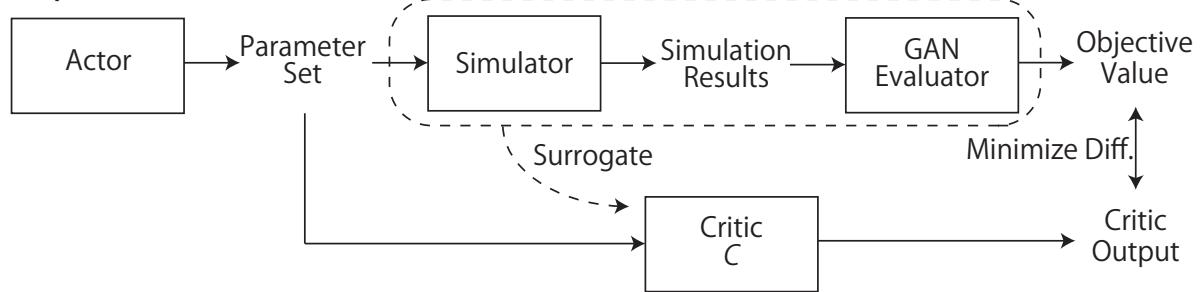


図 8.1: Actor-Critic によるパラメータチューニングにおける GAN Evaluator の挿入

提案手法の概要は、図 8.1 に示すとおりである。基本的に、図 7.1 における、Analyzer の部分を 4.2 章で提案した GAN Evaluator に置き換えることにより、シミュレーションの出力データに基づいた、妥当性の定量評価値を目的関数値とすることができる。

本章においては、7 章での結果を踏まえ、SAC ベースのチューニングモデルを採用した。そのため、GAN Evaluator と統合した手法の全体像は、図 8.2 の通りとなる。なお、チューニングのアルゴリズムは、7.2.2 章で説明したものと同様である。

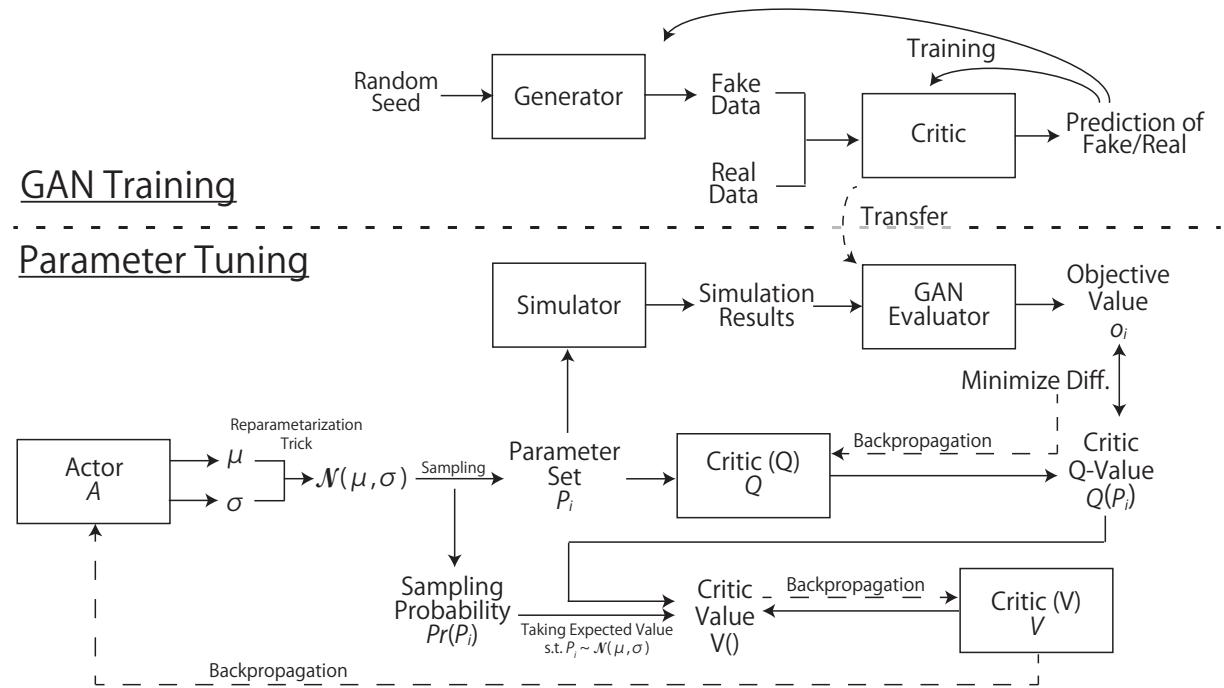


図 8.2: 実際の実装の概要

加えて、7章でSACベースのモデルの追加要素として用いていた、Seed Fixer (SF) および、Redundant Full Neural Network Actor (FNNA) は、同様に採用をした。さらに、7章と比べ、本章では、より高次元なパラメータチューニングに取り組むため、エージェント数もチューニングの対象のパラメータとした。エージェント数は常に正の整数を取るため、連続値でない。ゆえに、SACのSquashed Gaussian Policyを用いることができない。そこで、本章においては、Squashed Gaussian Policyの代わりに、7章でDDPGベースのモデルに適用していた、Action Converter (AC) を採用する。なお、ACの変換関数については、実験設定の章で後述する。

さらに、GAN Evaluatorが非常に計算コストが大きく、一度での計算効率を上げる必要がある。そのため、Actorからパラメータをサンプリングすることを10回行い、10個のパラメータセットに対して同時に計算を行うことで、GAN EvaluatorにおけるGPU演算の効率化を図った。つまり、Actorのアップデートの頻度が1/10に落ちるという事である。

8.3 実験

8.3.1 タスク設定

実験には、マルチエージェントシミュレーションとして、人工市場シミュレーションを採用した。

シミュレーションモデル

4.3.2節で用いたものと同様の人工市場モデルを使用する。ただし、ここでは、多次元のパラメータチューニングをテストするために、7章の実験よりも、チューニング対象を増やす。なお、ベイズ推定ベースの手法を採用した4.4節での実験と比較を行うため、4.4節と同じ以下のパラメータをチューニングの対象とした。

- エージェント数 n_{agent}
- ファンダメンタルファクターの重み w_F
- チャート(トレンド)ファクターの重み w_C
- ノイズスケール σ
- 平均回帰時間定数の最小値 τ_{\min}^*
- 平均回帰時間定数の最大値 τ_{\max}^*
- Time Window Size の最小値 τ_{\min}

- Time Window Size の最大値 τ_{\max}
- 固定注文マージンの最小値 k_{\min}
- 固定注文マージンの最大値 k_{\max}

目的関数

GAN Evaluator としては、4.3.1節と同様に、Hinge Loss を採用した PGSGAN (PGSGAN-HL) を用いるため、目的関数の出力は、 $[-1, 1]$ となり、大きいほどより現実的であるという評価になる。そのため、目的関数の最大化問題として、パラメータチューニングを行う。

加えて、4.3.1節と同様に、3章で作成した10銘柄のGANを用いて、この10銘柄に対するGANのそれぞれをGAN evaluatorとして採用した場合の出力値の平均をとることとした。

8.3.2 モデルの実装

SACベースのパラメータチューニングモデル

原則として、7章と実装は同じである者の、パラメータの次元数が異なることと、タスクが異なることから、各種学習パラメータの調整は行った。詳細については、以下の通りです。

- Actor (FNNA): 4層のMLPでからなる構造をもつ。隠れ層は100次元であり、出力は μ と σ に対応する10次元の出力2個である。最終層を除くすべての層では、Layer NormalizationとReLUを用いる。FNNAを用いるため、入力としては、100次元のすべての要素が1のダミーベクトルを採用する。
- Q-net ($V(P_i)$): 100次元の隠れ層をもつ、4層のMLPである。入力としては、10次元のパラメータセットを受け取り、目的関数値の推定値を1次元で出力する。最終層を除くすべての層では、Layer NormalizationとReLUを用いる。
- V-net ($V()$. FNNAありの場合): 4層のMLPでからなる構造をもつ。隠れ層は100次元であり、出力は1次元($V()$)である。最終層を除くすべての層では、Layer NormalizationとReLUを用いる。FNNAを用いるため、入力としては、100次元のすべての要素が1のダミーベクトルを採用する。
- V-net ($V()$. FNNAなしの場合): $V()$ に対応する1次元の勾配付きパラメータを返す。
- SACのSoft Target Update Ratio: 0.5

- Actor の学習率: 10^{-5}
- Critic の学習率: 10^{-5}
- Batch Size: 100
- Experience Replay のバッファーサイズ: 1,000
- Optimizer: Adam
- 各試行におけるシミュレーションシードの数 (SF のシードパターン): 10

Action Converter

Action Converter は、以下の通り実装した。

- エージェント数: $n_{\text{agent}} = \lfloor 1000 \times (1 - \exp(-P_{1,t})) \rfloor$
- ファンダメンタルファクターの重み: $w_F = P_{2,t}$
- チャート(トレンド)ファクターの重み: $w_C = P_{3,t}$
- ノイズスケール: $\sigma = 10^{-P_{4,t}}$
- 平均回帰時間定数の最小値: $\tau_{\min}^* = 10^{\min(P_{5,t}, P_{6,t})}$
- 平均回帰時間定数の最大値: $\tau_{\max}^* = 10^{\max(P_{5,t}, P_{6,t})}$
- Time Window Size の最小値: $\tau_{\min} = 10^{\min(P_{7,t}, P_{8,t})}$
- Time Window Size の最大値: $\tau_{\max} = 10^{\max(P_{7,t}, P_{8,t})}$
- 固定注文マージンの最小値: $k_{\min} = 10^{-\max(P_{9,t}, P_{10,t})}$
- 固定注文マージンの最大値: $k_{\max} = 10^{-\min(P_{9,t}, P_{10,t})}$

ここで、 $P_t = (P_{1,t}, P_{2,t}, P_{3,t}, P_{4,t}, P_{5,t}, P_{6,t}, P_{7,t}, P_{8,t}, P_{9,t}, P_{10,t})$ は、Actor が出力するパラメータである。

ここで、この Action Converter の変換関数の設計は、上記のもの以外にもいくらでも考えられる。本章では、ひとまず、この関数設計論については論じないこととするが、上記の設定としたのには一定の意味がある。

まず、エージェント数に関しては、エージェント数を増やしすぎると、計算にかかる時間が増加し、現実的でないことと、マルチエージェントシミュレーションにおいては、すべての実際の市場参加者を模倣することは現実的ではないことから、一定の抽象化を行うのが一般であるため、エージェント数を 1000 までに抑えたかったためである。また、ノ

イズスケール, 平均回帰時間定数の最小値と最大値, Time Window Size の最小値と最大値, 固定注文マージンの最小値と最大値は, 4.4 節での設定を参考に, 関数を設計した. なお, 各種最小値と最大値のパラメータ設定において, 最小値が最大値を下回る必要があり, この実現においては, \min 関数と \max 関数を採用した. ここで, 簡易的な検討を行うために, $a \leq b$ を満たす Action Converter を考える.

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \min(x, y) \\ \max(x, y) \end{bmatrix} \quad (8.1)$$

と定義した場合に,

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} a & b \end{bmatrix} = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & (x \leq y) \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & (x \geq y) \end{cases} \quad (8.2)$$

となり, 2つのパラメータの大小に関係なく, 勾配を直接的に逆伝播可能である.

一方で, 他に考えられる実装として,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x \\ x + y \end{bmatrix} \quad (8.3)$$

が考えられる. しかしながら, この y に対しては, 非負制約をかけなければならず, 二重に関数変形を行わなければならず, 効率が悪くなる. 実際に, 予備実験で, この後者の実装の実験をしていたが, あまりうまく行かなかったため, 前者の実装を採用することとした.

8.3.3 比較評価

比較評価実験として, 1,000 epochs の学習をした結果獲得されたパラメータに対する, GAN Evaluator の評価値を, 学習とは別のシミュレーションシードを用いて, 10 試行平均で計算した.

この結果を, 4.5 章の結果と比較することにより, 従来のベイズ推定とのパフォーマンス比較を行う.

なお, 8.3.2 節に示した SF のシードパターンが 10 であること, 8.2 節の末尾で説明した Actor によるパラメータサンプリングの 10 倍高頻度化, そして, 1,000 epochs であることを考慮すると, 全部で, 100,000 回のシミュレーションを要する. そのため, 300,000 回のシミュレーションで最終的なチューニングパラメータを獲得した 4.5 章よりはシミュレーション試行数が少ない. そこで, 4.5 章の 1 ステップのみでチューニングした場合の結果とも比較することで, 提案手法の性能について比較評価を行うこととする.

8.4 結果と考察

チューニングの結果、以下のパラメータを獲得した。

- エージェント数： $n_{\text{agent}} = 997$
- ファンダメンタルファクターの重み： $w_F = 4.32$
- チャート(トレンド)ファクターの重み： $w_C = 0.503$
- ノイズスケール： $\sigma = 1.17 \times 10^{-5}$
- 平均回帰時間定数の最小値： $\tau_{\min}^* = 492$
- 平均回帰時間定数の最大値： $\tau_{\max}^* = 132000$
- Time Window Size の最小値： $\tau_{\min} = 9.20$
- Time Window Size の最大値： $\tau_{\max} = 5840$
- 固定注文マージンの最小値： $k_{\min} = 0.0373$
- 固定注文マージンの最大値： $k_{\max} = 0.112$

このパラメータに対する、GAN Evaluatorによる評価値は 0.960289 であった。

4.5 節での、ベイズ推定ベースの TPE を使用した場合のチューニング結果の 0.96524 よりは若干低い結果であり、同じシミュレーション回数で実験をしている、1段階目の結果の 0.961491 と比較しても、若干悪い結果ではある。

図 8.3–8.14 は、学習の 1,000 epochs における、各パラメータの変化と、スコアを示している。ただし、これらの結果は、あくまで学習時のスコアであるため、最終的な評価値とは異なる、In-sample な評価値であることに注意されたい。

図 8.3 では、各 epoch までの最高のパラメータを用いたときの GAN Evaluator の結果を示している。この結果によると、おおむね 400 epochs ですでに学習が収束しているよう見える。そのため、この 3 倍の学習ステップを設けても、なかなか収束判定が難しかった 4.5 節でのベイズ推定ベースの TPE と比べると、扱いが比較的楽であると言える。

さらに、図 8.4 では、各 epoch で探索されたパラメータに対する評価値の平均値の変化を示している。この結果を見ると、決して、平均が 1 に寄っていくことはなく、比較的広範なパラメータを探索しており、その結果として、平均が 0 付近に近づいていることと、学習終盤になっても、探索を継続していることが確認できる。これは、SAC の強い探索力特有の結果ではないかと考えられる。

図 8.5 ではエージェント数に関するチューニングの遷移を示している。これを見ると、序盤から順調にエージェント数を増やす方向に学習していることが確認できる。そして、最終的には、1000 未満という制約の目いっぱいまで探索領域が寄って行っていることが

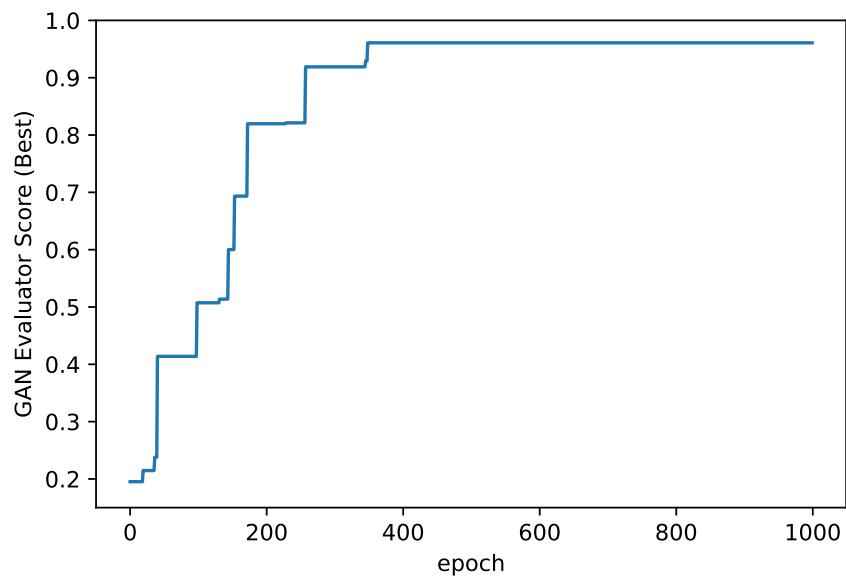


図 8.3: 最高パラメータに対する評価値の変化(学習時)

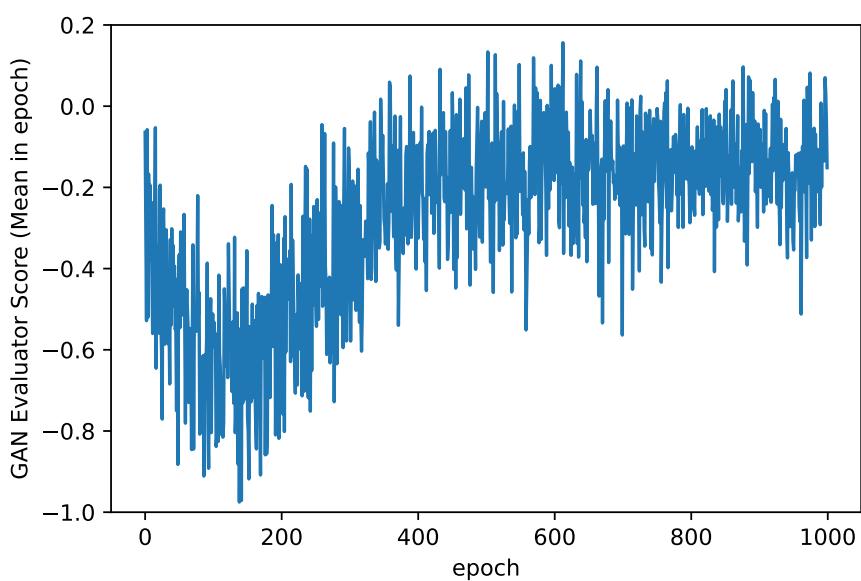


図 8.4: 各 epoch で探索されたパラメータに対する評価値の平均値の変化

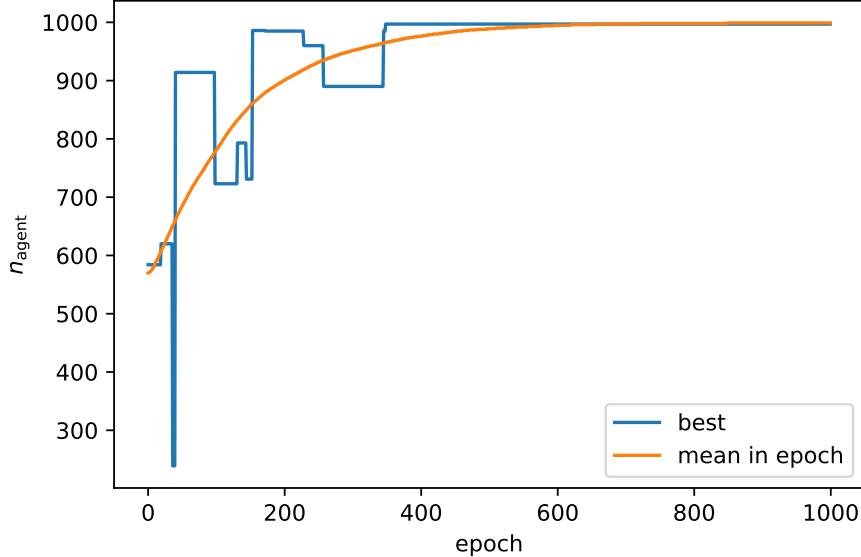


図 8.5: 各 epoch までの評価値が最高であったパラメータセットの n_{agent} (青) および各 epoch で探索された n_{agent} の平均値(オレンジ)の変化

確認できる。そのため、本来的にはエージェント数は 1000 より大きい方がより現実的なシミュレーションであることが示唆されるが、これは、実際の計算リソースを考慮に入れると、限界があるので、やむを得ない結果となっているともいえる。

図 8.6 では、ファンダメンタルファクターの重みに関するチューニング結果を示している。この結果によると、最適パラメータとして、 w_F は 4 付近で最終的に安定しているにも関わらず、オレンジの線が示す探索パラメータの平均値は、最後まで上昇している。これは、より大きな値に対して探索領域を広げて最適値がないかを探索していると推定できる。

図 8.7 では、チャートファクターの重みに関する結果を示している。こちらも、 w_F の場合と同様に、最適パラメータが安定化してもなお、探索の領域を上方に広げていると推測できる結果となっている。

図 8.8 では、ノイズスケールのチューニング結果の遷移を示している。これによると、初期値から徐々に小さい値にパラメータが遷移していき、最適パラメータが安定してもなお、それよりも小さい値を探索していることが確認できる。

図 8.9 および図 8.10 では、平均回帰時間定数の最小値と最大値のチューニングの遷移を示している。最小値が最大値を上回らないという設計をしているため、似たようなグラフであるものの、その制約を満たすように遷移をしていることが確認できる。また、最適パラメータが安定後も、より上方の値の探索を行っていることが確認できる。

図 8.11 および図 8.12 では、Time Window Size の最小値と最大値のチューニングの推

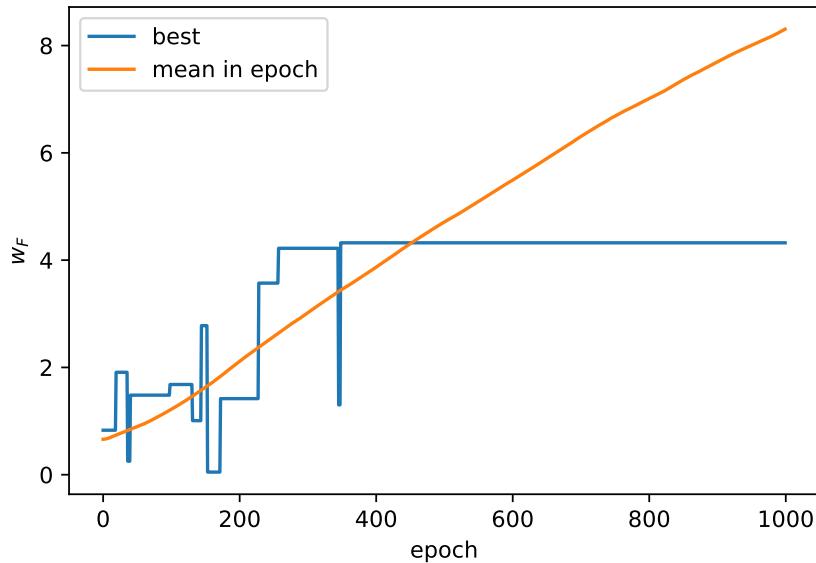


図 8.6: 各 epoch までの評価値が最高であったパラメータセットの w_F (青) および各 epoch で探索された w_F の平均値(オレンジ)の変化

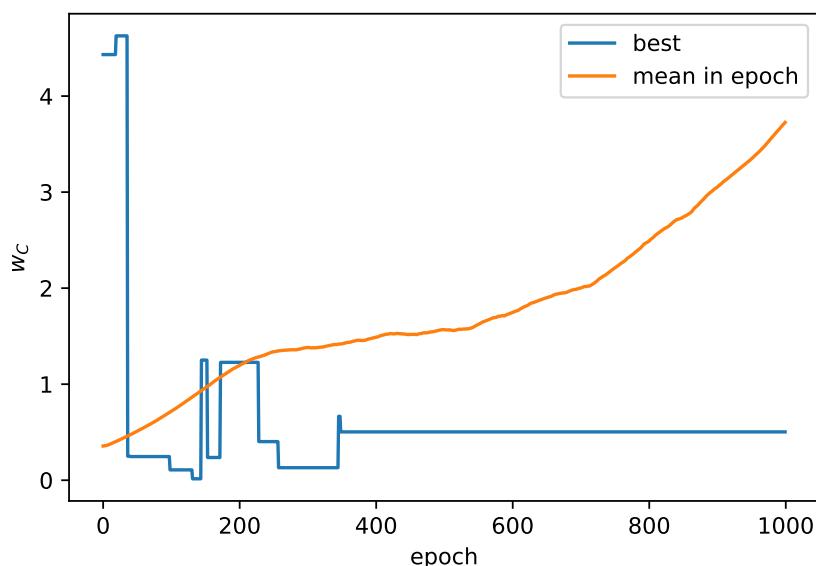


図 8.7: 各 epoch までの評価値が最高であったパラメータセットの w_C (青) および各 epoch で探索された w_C の平均値(オレンジ)の変化

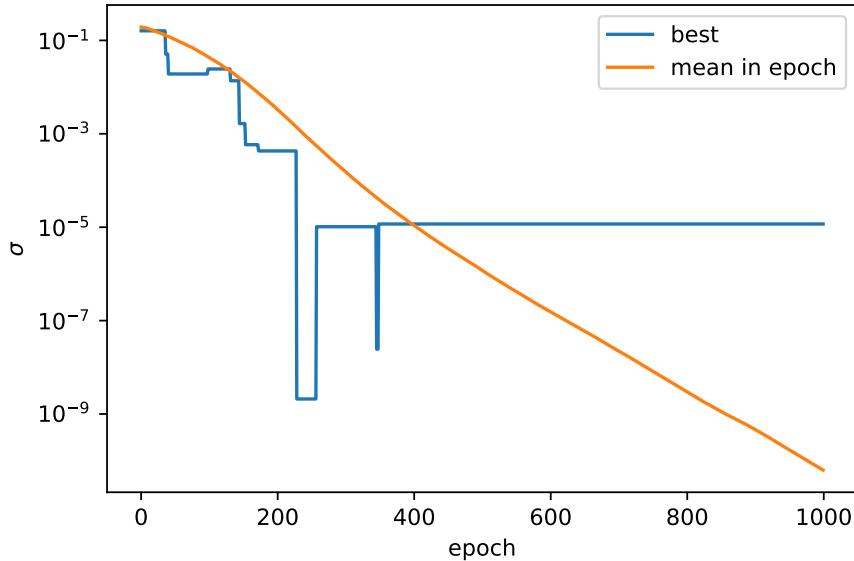


図 8.8: 各 epoch までの評価値が最高であったパラメータセットの σ (青) および各 epoch で探索された σ の平均値(オレンジ)の変化

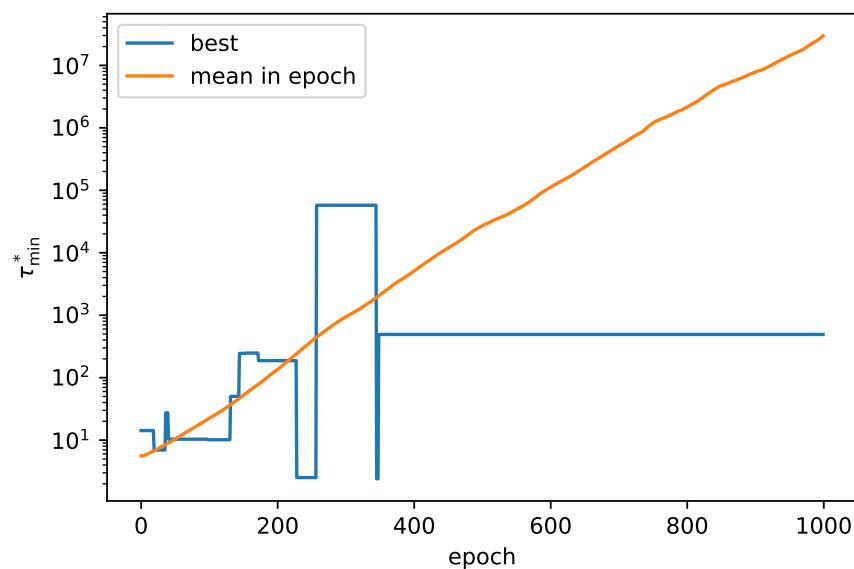


図 8.9: 各 epoch までの評価値が最高であったパラメータセットの τ_{\min}^* (青) および各 epoch で探索された τ_{\min}^* の平均値(オレンジ)の変化

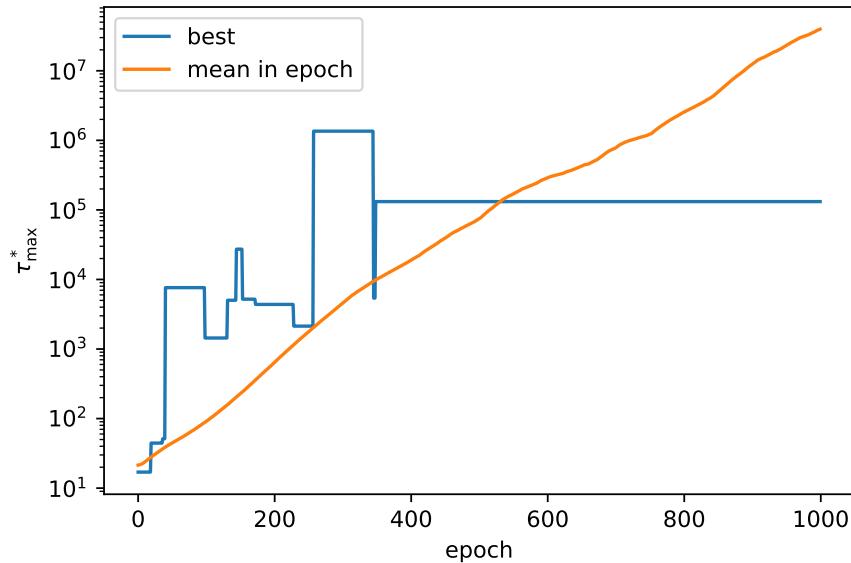


図 8.10: 各 epoch までの評価値が最高であったパラメータセットの τ_{\max}^* (青) および各 epoch で探索された τ_{\max}^* の平均値(オレンジ)の変化

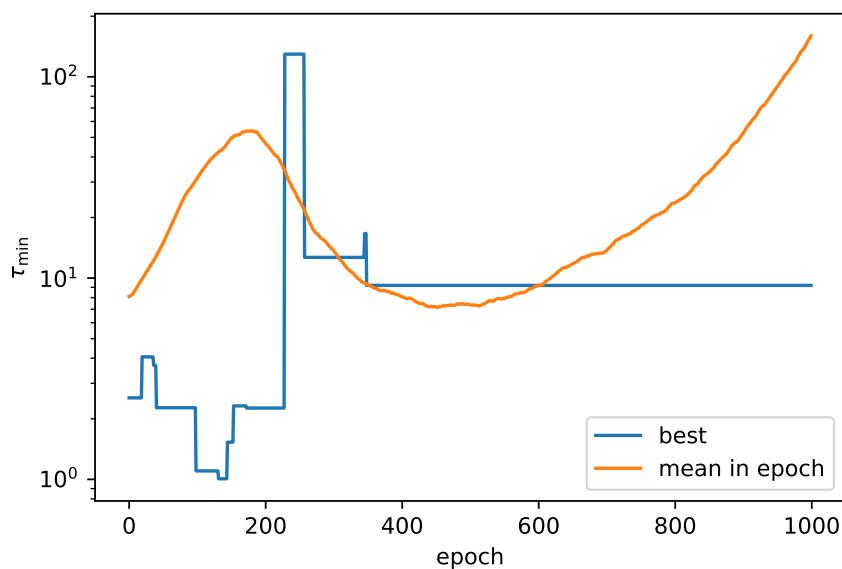


図 8.11: 各 epoch までの評価値が最高であったパラメータセットの τ_{\min}^* (青) および各 epoch で探索された τ_{\min}^* の平均値(オレンジ)の変化

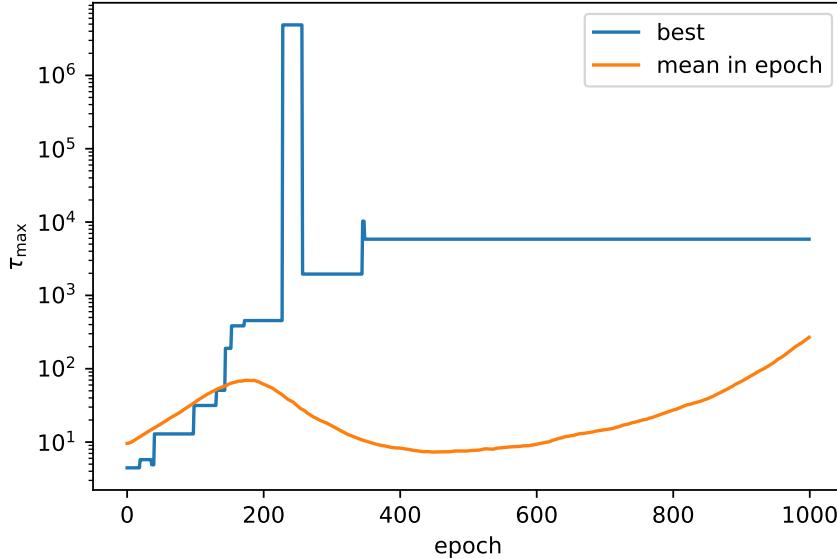


図 8.12: 各 epoch までの評価値が最高であったパラメータセットの τ_{\max} (青) および各 epoch で探索された τ_{\max} の平均値(オレンジ)の変化

移を示している。こちらも、最小値が最大値よりも小さいという制約の結果、部分的に似たような変化を起こしている部分が確認できるのとともに、各 epoch での探索パラメータの平均の推移の形が非常に似たものとなっている。

図 8.13 および図 8.14 では、固定注文マージンの最小値と最大値のチューニングの遷移を示している。これらは、400 epochs 程度で収束しているもの、その後も、より小さいパラメータを探している様子が顕著に確認できる。

ここまで結果を踏まえると、ベイズ推定ベースの手法には若干及ばない結果になっているものの、多次元のパラメータチューニングにおいて、適切な形でパラメータ探索が行われているように見受けられる。勾配ベースの手法であるために、探索点の集合として最適値を探索するベイズ推定の手法よりは、連続的に滑らかに探索を行っている可能性があり、場合によっては、これが強みになる可能性がある。

加えて、今回の実験では、一部パラメータ(特にエージェント数)に制約を課して実験を行ったが、技術上は、有限ではない空間に対して探索を行うことが可能である。この点は、探索レンジを決めなければならない TPE に比べると、有意性があるともいえる。

本章の実験においては、Action Converter で使用する関数については、深く比較検討を行わなかったが、この関数の設計論を構築できれば、提案手法の適用可能性はより広くなるのではないかと考える。加えて、本章の実験では、8.3.2 節などで示した多数のハイパーパラメータのチューニングはできていない。これは、原状で、計算リソース上の限界があるためであるが、これらのチューニングを行うことで、4 章の結果を上回るチューニングを行うことができる可能性もあると考えている。そのため、今後の課題として、さら

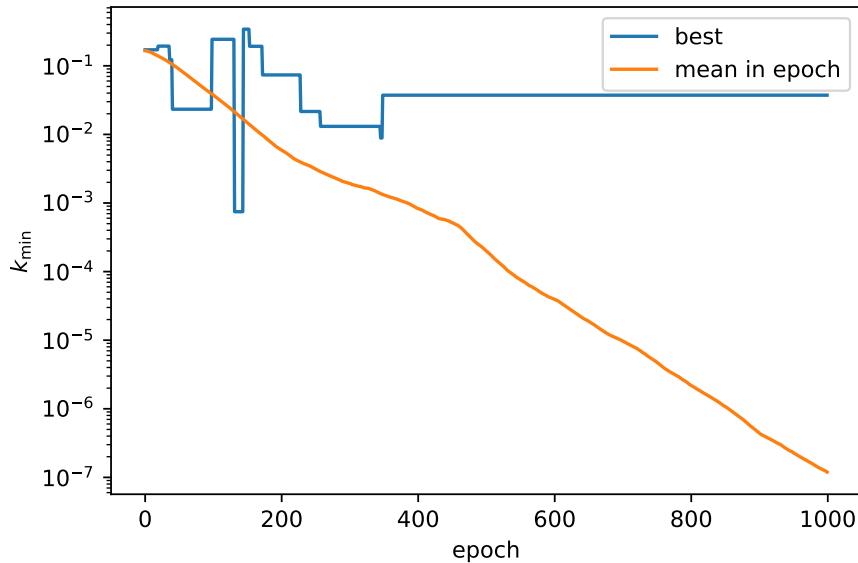


図 8.13: 各 epoch までの評価値が最高であったパラメータセットの k_{\min} (青) および各 epoch で探索された k_{\min} の平均値(オレンジ)の変化

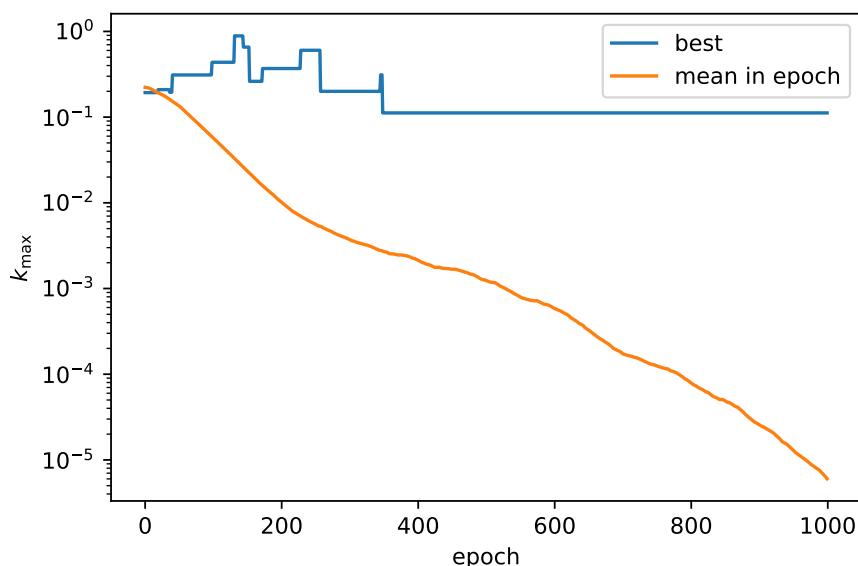


図 8.14: 各 epoch までの評価値が最高であったパラメータセットの k_{\max} (青) および各 epoch で探索された k_{\max} の平均値(オレンジ)の変化

なる計算効率の向上があげられる。

本章においては、4章で示した、GANを用いたシミュレーションの妥当性評価機構である、GAN Evaluatorと、7章で示した、強化学習機構を用いた、シミュレーションのパラメータチューニング手法の融合を行った。この融合という観点では、ベイズ推定ベースのチューニングと大差のない結果を達成することに成功しており、さらなる個別技術の向上により、マルチエージェントシミュレーションにおける、現実性の向上が期待できると考える。

8.5 本章のまとめ

本章では、4章で示した、GANを用いたシミュレーションの妥当性評価機構である、GAN Evaluatorと、7章で示した、強化学習機構を用いた、シミュレーションのパラメータチューニング手法の融合を行った。これにより、現実的な結果を再現できるシミュレーションパラメータの探索を実現することができる。実験の結果、この融合により、ベイズ推定とほぼ同等で、現実的であると考えられるシミュレーションを再現できるパラメータを獲得できることができた。これにより、マルチエージェントシミュレーションにおいて、現実的なシミュレーションであるかどうかという点について、定量的に評価を行い、その評価を最大化するような最適パラメータを探索する、深層学習ベースの手法の確立を達成した。

第V部

まとめ

第9章 全体を通した考察と今後の課題

まず、各章についてまとめる。

3章では、金融市場にフォーカスし、金融市場のデータを用いて、PGSGANと呼ばれる、注文生成にサンプリングプロセスを用いた手法を提案し、注文生成において、先行研究より高いパフォーマンスを達成した。しかしながら、本博士論文においてはGANの生成自体は目的ではなく、手段であり、その作成したPGSGANを用いて、シミュレーションの定量評価手法を4章で提案した。このシミュレーションの定量評価手法は、従来のStylized Factsに基づく、定性的な評価基準と整合性がそれなりに取れるものであることが実験の結果確認された。

そして、5章では、人工市場データマイニングプラットフォームという、人工市場内でデータマイニングの精度評価を行うスキームを提案し、その実例として、6章では、金融市場における注文の同時性がデータマイニング手法に与える影響について検証した。これらの研究を通じて、人工市場データマイニングプラットフォームの有効性について示した。

また、7章では、強化学習手法を活用した、シミュレーションのパラメータチューニング手法について提案した。特に、SACベースの手法に、追加提案した学習を安定化させる追加要素を用いることが有効であることが確認できた。

そして、最後に、4章のシミュレーションの定量評価手法と7章の強化学習手法を活用した、シミュレーションのパラメータチューニング手法を融合させた手法を提案・検証した。この結果、深層学習を用いて、マルチエージェントシミュレーションの妥当性評価と、妥当性のあるパラメータ探索を可能にすることができるということを示すことができた。

まず、これらのすべての結果を踏まえて言えることとして、本研究を通じて、当初の研究背景であるところの、シミュレーションとデータマイニングの融合は図ることができたのではないかと考えられる。まず、5章と6章では、シミュレーションからデータマイニング方向へのメリットの享受のある技術を開発した。それ以外の章では、データマイニングによるメリットをシミュレーションに取り込む技術を開発した。本論文においては、後者のデータマイニングからシミュレーション方向へのメリットの享受が本題ではあるが、この点に関しては、現象理解のツールとしてのシミュレーションというメリットをつぶすことなく、技術開発ができた点については大きいと考えている。

本論文においては、深層学習手法を多用したが、改めて、深層学習の強みが明らかになったと考えている。深層強化学習や、大規模モデルにおける各種タスクにおける性能の高さなどは、2章で確認した多数の先行研究などから明らかであったものの、これまでに行われてこなかった、シミュレーションとデータマイニングの融合という観点でも、非常

に有効な手法となっていたと考える。

一方で、本論文で扱い切れていない点も複数存在する。シミュレーションのメリットである、未知の事象にどの程度対応可能なシミュレーションであるのかについては、本論文では検討できていない。ただし、これは、そもそも未知の事象への対応という点を評価が困難であるため、決して検討が容易なものではないと考えている。そのため、未知の事象への対応可能性という点の定義から含めて、今後の課題であると考える。

また、本論文においては、比較的、GANの性能が良いために、4章のシミュレーションの定量評価手法がうまく行った可能性は否定できない。そのため、今後、より広く活用されるためには、どの程度、GANの性能が要求されるのか、つまり、性能の悪いGANでも、ある程度シミュレーションの定量評価に使用可能であるのか、などの点については、さらなる議論が必要であると考えられる。

さらに、本論文で使用したシミュレーションには限りがある点も、今後の課題である。本論文では、原則として、シミュレーションのモデリングに関しての議論を回避するために、先行研究に基づいた人工市場のシミュレーションを採用した。しかしながら、実際には、どのモデルを採用するかや、他の分野のシミュレーションでも適用可能であるかという観点に関しては、議論の余地がある。この点についても、今後の課題としたい。

第10章 まとめ

金融市場において、マルチエージェントシミュレーションとデータマイニングはとても有用な技術である。マルチエージェントシミュレーションとは、コンピューター上に仮想の世界と仮想のエージェントを作成し、エージェントのミクロな行動の積み上げとして世界全体の動きを再現する手法であり、複雑系の分析に有用である。一方で、データマイニングは、世の中に存在するデータを用いて、様々な知識や現象をとらえようという取り組みである。特に、金融市場においては、マルチエージェントシミュレーションは、実市場で検証できないシナリオの検証などで有用であり、データマイニングは、利益を追求するなどの場面で多く使われてきている。

これらのシミュレーションとデータマイニングのメリットデメリットは相補的な関係になっていると考える。たとえば、データマイニングはサンプルに存在しないデータに弱い一方で、シミュレーションは未知のシチュエーションに対応できるという強みを持つ。また、データマイニングはブラックボックス化しやすい一方で、現象理解しやすいという強みを持つ。シミュレーションはモデリングが人間のセンスに依存しやすい一方で、データマイニングはデータという客観的なものに基づいてモデルを構築する。また、シミュレーションは出力データに現実味がないとされる一方で、データマイニングは現実の非常に多くのデータを使う。これらは、相反する特徴としてとらえられてきた結果、シミュレーションとデータマイニングの融合はなかなか行われてこなかった。しかしながら、相反すると考えるのではなく、補完的な存在であると考えることによって、その双方の強みを活用したソリューションを提案できるのではないかだろうかと考え、本論文においては、マルチエージェントシミュレーションとデータマイニングを融合させ、それら双方の強みを生かす手法の構築を行った。

まず、金融市場のデータを用いて、PGSGANと呼ばれる、注文生成にサンプリングプロセスを用いた手法を提案し、注文生成において、先行研究より高いパフォーマンスを達成した。そのうえで、その作成したPGSGANを用いて、シミュレーションの定量評価手法を提案し、従来のStylized Factsに基づく、定性的な評価基準と整合性が取れるものであることを示した。さらに、人工市場データマイニングプラットフォームという、人工市場内でデータマイニングの精度評価を行うスキームを提案し、その実例として、金融市場における注文の同時性がデータマイニング手法に与える影響について検証するとともに、このプラットフォームの有効性を示した。また、強化学習手法を活用した、シミュレーションのパラメータチューニング手法について提案し、シミュレーションの定量評価手法と融合させた手法を提案・検証した。この結果、深層学習を用いて、マルチエージェ

ントシミュレーションの妥当性評価と、妥当性のあるパラメータ探索を可能にすることができるということを示すことができた。これらの研究を通じて、シミュレーションとデータマイニングの融合は図ることができた。

参考文献

- [1] T. Mizuta, S. Kosugi, T. Kusumoto, W. Matsumoto, K. Izumi, I. Yagi, and S. Yoshimura, “Effects of Price Regulations and Dark Pools on Financial Market Stability: An Investigation by Multiagent Simulations,” *Intelligent Systems in Accounting, Finance and Management*, vol. 23, no. 1-2, pp. 97–120, 2016.
- [2] M. Hirano, K. Izumi, T. Shimada, H. Matsushima, and H. Sakaji, “Impact Analysis of Financial Regulation on Multi-Asset Markets Using Artificial Market Simulations,” *Journal of Risk and Financial Management*, vol. 13, no. 4, p. 75, 2020.
- [3] J. Wang, T. Sun, B. Liu, Y. Cao, and H. Zhu, “CLVSA: A Convolutional LSTM Based Variational Sequence-to-Sequence Model with Attention for Predicting Trends of Financial Markets,” in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 3705–3711, 2019.
- [4] D. Tashiro, H. Matsushima, K. Izumi, and H. Sakaji, “Encoding of High-frequency Order Information and Prediction of Short-term Stock Price by Deep Learning,” *Quantitative Finance*, vol. 19, no. 9, pp. 1499–1506, 2019.
- [5] R. Axelrod, “The complexity of cooperation,” in *The Complexity of Cooperation*, Princeton university press, 1997.
- [6] B. Edmonds and S. Moss, “From KISS to KIDS—an ‘anti-simplistic’ modelling approach,” in *Proceedings of International Workshop on Multi-agent Systems and Agent-based Simulation*, pp. 130–144, Springer, 2004.
- [7] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-scale Image Recognition,” *arXiv*, 2014.
- [8] M. Tan and Q. Le, “Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks,” in *Proceedings of International Conference on Machine Learning*, pp. 6105–6114, 2019.
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778, IEEE Computer Society, 2016.

- [10] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A Convnet for the 2020s,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.
- [11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” *arXiv*, 2020.
- [12] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.
- [13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is All You Need,” in *Advances in Neural Information Processing Systems*, pp. 5999–6009, 2017.
- [14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, June 2019.
- [15] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” <https://cdn.openai.com/research-covers/language-unsupervised/language-understanding-paper.pdf>.
- [16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multi-task Learners,” 2019. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
- [17] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.
- [18] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized Autoregressive Pretraining for Language Understanding,” in *Advances in Neural Information Processing Systems*, vol. 32, pp. 5753–5763, 2019.

- [19] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations,” in *Proceedings of 8th International Conference on Learning Representations*, 2019.
- [20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, and P. G. Allen, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” *arXiv*, 2019.
- [21] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators,” in *Proceedings of 8th International Conference on Learning Representations*, 2020.
- [22] M. Suzuki, H. Sakaji, M. Hirano, and K. Izumi, “Construction and Validation of a Pre-Trained Language Model Using Financial Documents,” in *Proceedings of JSAI Special Interest Group on Financial Infomatics (SIG-FIN) 27*, pp. 5–10, 2021.
- [23] A. Fisher, C. Rudin, and F. Dominici, “All Models are Wrong, but Many are Useful: Learning a Variable’s Importance by Studying an Entire Class of Prediction Models Simultaneously,” *Journal of Machine Learning Research*, vol. 20, no. 177, pp. 1–81, 2019.
- [24] Q. Zhao and T. Hastie, “Causal Interpretations of Black-box Models,” *Journal of Business & Economic Statistics*, vol. 39, no. 1, pp. 272–281, 2021.
- [25] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [26] M. Hirano, K. Izumi, and H. Sakaji, “Implementation of Actual Data for Artificial Market Simulation,” in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 1624–1626, 2022.
- [27] M. Hirano, K. Izumi, and H. Sakaji, “Data-driven Agent Design for Artificial Market Simulation,” in *Proceedings of the 36th Annual Conference of the Japanese Society for Artificial Intelligence*, pp. 2S4IS2b01–2S4IS2b01, 2022.
- [28] M. Hirano, H. Sakaji, and K. Izumi, “Policy Gradient Stock GAN for Realistic Discrete Order Data Generation in Financial Markets,” *arXiv*, 2022.
- [29] M. Hirano and K. Izumi, “Quantitative Tuning of Artificial Market Simulation using Generative Adversarial Network,” in *Proceedings of the 6th IEEE International Conference on Agents*, pp. 12–17, 2022.

- [30] M. Hirano, H. Sakaji, and K. Izumi, “Concept and Practice of Artificial Market Data Mining Platform,” in *Proceedings of 2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics*, pp. 1–10, 2022.
- [31] M. Hirano and K. Izumi, “Does Order Simultaneity Affect the Data Mining Task in Financial Markets? – Effect Analysis of Order Simultaneity using Artificial Market,” in *Proceedings of the 24th International Conference on Principles and Practice of Multi-Agent Systems*, pp. 297–313, 2022.
- [32] M. Hirano and K. Izumi, “Parameter Tuning Method for Multi-agent Simulation using Reinforcement Learning,” in *Proceedings of the 9th International Conference on Behavioral and Social Computing*, pp. 1–7, 2022.
- [33] M. Hirano and K. Izumi, “Efficient Parameter Tuning for Multi-agent Simulation Using Deep Reinforcement Learning,” in *Proceedings of 13th IIAI International Congress on Advanced Applied Informatics*, pp. 130–137, 2022.
- [34] M. Hirano and K. Izumi, “Quantitative Evaluation of Multi-agent Simulation using Generative Adversarial Network – An Alternative of Qualitative Evaluation for Artificial Market Simulation,” in *Proceedings of the 37th Annual Conference of the Japanese Society for Artificial Intelligence*, 2023.
- [35] M. Hirano and K. Izumi, “Neural-network-based Parameter Tuning for Multi-agent Simulation using Deep Reinforcement Learning,” *World Wide Web*, 2023.
- [36] T. C. Schelling, “Models of Segregation,” *The American Economic Review*, vol. 59, no. 2, pp. 488–493, 1969.
- [37] R. Axelrod, “Effective Choice in the Prisoner’s Dilemma,” *Journal of Conflict Resolution*, vol. 24, no. 1, pp. 3–25, 1980.
- [38] R. Axelrod, “More Effective Choice in the Prisoner’s Dilemma,” *Journal of Conflict Resolution*, vol. 24, no. 3, pp. 379–403, 1980.
- [39] J. M. Epstein and R. Axtell, *Growing Artificial Societies: Social Science from the Bottom up*. Brookings Institution Press, 1996.
- [40] T. Lux and M. Marchesi, “Scaling and Criticality in a Stochastic Multi-agent Model of a Financial Market,” *Nature*, vol. 397, no. 6719, pp. 498–500, 1999.
- [41] M. Sajjad, K. Singh, E. Paik, and C. W. Ahn, “A Data-driven Approach for Agent-based Modeling: Simulating the Dynamics of Family Formation,” *Journal of Artificial Societies and Social Simulation*, vol. 19, no. 1, 2016.

- [42] Y. Nonaka, M. Onishi, T. Yamashita, T. Okada, A. Shimada, and R. I. Taniguchi, “Walking Velocity Model for Accurate and Massive Pedestrian Simulator,” *IEEJ Transactions on Electronics, Information and Systems*, vol. 133, no. 9, 2013.
- [43] K. Braun-Munzinger, Z. Liu, and A. E. Turrell, “An Agent-based Model of Corporate Bond Trading,” *Quantitative Finance*, vol. 18, no. 4, pp. 591–608, 2018.
- [44] S. Kurahashi, “Estimating Effectiveness of Preventing Measures for 2019 Novel Coronavirus Diseases (COVID-19),” *Proceeding of 2020 9th International Congress on Advanced Applied Informatics*, pp. 487–492, 2020.
- [45] S. M. Edmonds and Bruce, “Towards Good Social Science,” *Journal of Artificial Societies and Social Simulation*, vol. 8, no. 4, 2005. <https://www.jasss.org/8/4/13.html>.
- [46] J. D. Farmer and D. Foley, “The Economy Needs Agent-based Modelling,” *Nature*, vol. 460, no. 7256, pp. 685–686, 2009.
- [47] S. Battiston, J. D. Farmer, A. Flache, D. Garlaschelli, A. G. Haldane, H. Heesterbeek, C. Hommes, C. Jaeger, R. May, and M. Scheffer, “Complexity Theory and Financial Regulation: Economic Policy Needs Interdisciplinary Network Analysis and Behavioral Modeling,” *Science*, vol. 351, no. 6275, pp. 818–819, 2016.
- [48] 寺野隆雄, “エージェントベースモデリング: Kiss 原理を超えて (<特集>複雑系と集合知),” *人工知能*, vol. 18, no. 6, pp. 710–715, 2003.
- [49] T. Mizuta, “An Agent-based Model for Designing a Financial Market that Works Well,” in *Proceedings of 2020 IEEE Symposium Series on Computational Intelligence*, 2019.
- [50] J.-C. Trichet, “Reflections on the Nature of Monetary Policy Non-standard Measures and Financial Study,” in *Approaches to Monetary Policy Revisited - Lessons from the Crisis*, pp. 12–22, European Central Bank, 2011.
- [51] R. M. Bookstaber, *The End of Theory : Financial Crises, the Failure of Economics, and the Sweep of Human Interaction*. Princeton University Press, 2017.
- [52] W. Cui and A. Brabazon, “An agent-based modeling approach to study price impact,” in *Proceedings of 2012 IEEE Conference on Computational Intelligence for Financial Engineering and Economics*, pp. 241–248, 2012.
- [53] T. Torii, K. Izumi, and K. Yamada, “Shock Transfer by Arbitrage Trading: Analysis using Multi-asset Artificial Market,” *Evolutionary and Institutional Economics Review*, vol. 12, no. 2, pp. 395–412, 2015.

- [54] S. J. Leal and M. Napoletano, “Market Stability vs. Market Resilience: Regulatory Policies Experiments in an Agent-based Model with Low- and High-frequency Trading,” *Journal of Economic Behavior and Organization*, vol. 157, pp. 15–41, 2019.
- [55] M. Paddrik, R. Hayes, A. Todd, S. Yang, P. Beling, and W. Scherer, “An Agent Based Model of the E-Mini S&P 500 Applied to Flash Crash Analysis,” in *Proceedings of 2012 IEEE Conference on Computational Intelligence for Financial Engineering and Economics*, pp. 257–264, 2012.
- [56] T. Torii, T. Kamada, K. Izumi, and K. Yamada, “Platform Design for Large-scale Artificial Market Simulation and Preliminary Evaluation on the K Computer,” *Artificial Life and Robotics*, vol. 22, no. 3, pp. 301–307, 2017.
- [57] T. Torii, K. Izumi, T. Kamada, H. Yonenoh, D. Fujishima, I. Matsuura, M. Hirano, and T. Takahashi, “Plham: Platform for Large-scale and High-frequency Artificial Market,” 2016. <https://github.com/plham/plham>.
- [58] T. Torii, K. Izumi, T. Kamada, H. Yonenoh, D. Fujishima, I. Matsuura, M. Hirano, T. Takahashi, and P. Finnerty, “PlhamJ,” 2019. <https://github.com/plham/plhamJ>.
- [59] M. Hirano and R. Takata, “PAMS: Platform for Artificial Market Simulations,” 2022. <https://github.com/masanorihirano/pams>.
- [60] H. Sato, Y. Koyama, K. Kurumatani, Y. Shiozawa, and H. Deguchi, “U-mart: A test bed for interdisciplinary research into agent-based artificial markets,” in *Evolutionary Controversies in Economics*, pp. 179–190, Springer, 2001.
- [61] W. B. Arthur, J. H. Holland, B. LeBaron, R. Palmer, and P. Tayler, “Asset Pricing under Endogenous Expectations in an Artificial Stock Market,” *The Economy as an Evolving Complex System II*, pp. 15–44, 1997.
- [62] D. Byrd, M. Hybinette, T. Hybinette Balch, and J. Morgan, “ABIDES: Towards High-Fidelity Multi-Agent Market Simulation,” in *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, vol. 12, 2020.
- [63] Y. Murase, T. Uchitane, and N. Ito, “A Tool for Parameter-space Explorations,” *Physics Procedia*, vol. 57, no. C, pp. 73–76, 2014.
- [64] Y. Murase, H. Matsushima, I. Noda, and T. Kamada, “CARAVAN: A Framework for Comprehensive Simulations on Massive Parallel Machines,” in *Proceedings of International Workshop on Massively Multiagent Systems*, pp. 130–143, 2019.

- [65] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, “Multiobjective Tree-structured Parzen Estimator for Computationally Expensive Optimization Problems,” in *Proceedings of 2020 Genetic and Evolutionary Computation Conference*, pp. 533–541, Association for Computing Machinery, 2020.
- [66] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” in *Proceedings of the 25th International Conference on Knowledge Discovery & Data mining*, pp. 2623–2631, 2019.
- [67] J. J. Grefenstette and J. M. Fitzpatrick, “Genetic Search with Approximate Function Evaluation,” in *Proceedings of the 1st International Conference on Genetic Algorithms*, p. 112–120, 1985.
- [68] G. Schneider, J. Schuchhardt, and P. Wrede, “Artificial Neural Networks and Simulated Molecular Evolution are Potential Tools for Sequence-oriented Protein Design,” *Bioinformatics*, vol. 10, no. 6, pp. 635–645, 1994.
- [69] D. Yang and S. J. Flockton, “Evolutionary Algorithms with a Coarse-to-fine Function Smoothing,” in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, vol. 2, pp. 657–662, 1995.
- [70] A. Ratle, “Accelerating the Convergence of Evolutionary Algorithms by Fitness Landscape Approximation,” in *Proceedings of International Conference on Parallel Problem Solving from Nature*, pp. 87–96, 1998.
- [71] S. PIERRET and R. VAN DEN BRAEMBUSSCHE, “Turbomachinery Blade Design using a Navier-Stokes Solver and Artificial Neural Network,” *Journal of Turbomachinery*, vol. 121, no. 2, pp. 326–332, 1999.
- [72] Y. Yamashita, S. Shigenaka, D. Oba, and M. Onishi, “Estimation of Large-scale Multi Agent Simulation Results Using Neural Networks [in Japanese],” in *Proceedings of Japanese Special Interest Group on Society and Artificial Intelligence (SIG-SAI)*, vol. 2020, p. 05, 2020.
- [73] C. Angione, E. Silverman, and E. Yaneske, “Using Machine Learning as a Surrogate Model for Agent-based Simulations,” *PLOS ONE*, vol. 17, no. 2, p. e0263150, 2022.
- [74] K. Kim, “Financial Time Series Forecasting using Support Vector Machines,” *Neurocomputing*, vol. 55, pp. 307–319, 2003.

- [75] A. N. Kercheval and Y. Zhang, “Modelling High-frequency Limit Order Book Dynamics with Support Vector Machines,” *Quantitative Finance*, vol. 15, no. 8, pp. 1315–1329, 2015.
- [76] J. A. Sirignano, “Deep Learning for Limit Order Books,” *Quantitative Finance*, vol. 19, no. 4, pp. 549–570, 2019.
- [77] A. Tsantekidis, N. Passalis, A. Tefas, J. Kannaiainen, M. Gabbouj, and A. Iosifidis, “Using Deep Learning to Detect Price Change Indications in Financial Markets,” in *Proceedings of the 25th European Signal Processing Conference*, pp. 2580–2584, 2017.
- [78] M. F. Dixon, N. G. Polson, and V. O. Sokolov, “Deep Learning for Spatio - temporal Modeling: Dynamic Traffic Flows and High Frequency Trading,” *Quantitative Finance*, vol. 19, no. 4, pp. 549–570, 2019.
- [79] L. Zhang, C. Aggarwal, and G.-J. Qi, “Stock Price Prediction via Discovering Multi-Frequency Trading Patterns,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2141–2149, 2017.
- [80] D. Tashiro and K. Izumi, “Estimating Stock Orders using Deep Learning and High Frequency Data [in Japanese],” in *Proceedings of the 31nd Annual Conference of the Japanese Society for Artificial*, pp. 2D2–2, 2017.
- [81] Nanex, “Nanex - Market Crop Circle of the Day,” 2010.
<http://www.nanex.net/FlashCrash/CCircleDay.html>.
- [82] R. Cont, “Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues,” *Quantitative Finance*, vol. 1, no. 2, pp. 223–236, 2001.
- [83] B. Miyazaki, K. Izumi, F. Toriumi, and R. Takahashi, “Change Detection of Orders in Stock Markets Using a Gaussian Mixture Model,” *Intelligent Systems in Accounting, Finance and Management*, vol. 21, no. 3, pp. 169–191, 2014.
- [84] M. Hirano, H. Matsushima, K. Izumi, and H. Sakaji, “STBM : Stochastic Trading Behavior Model for Financial Markets Based on Long Short-Term Memory,” in *Proceedings of the 34th Annual Conference of the Japanese Society for Artificial Intelligence*, pp. 1K4-ES-2-04, 2020.
- [85] M. Hirano, K. Izumi, and H. Sakaji, “STBM+: Advanced Stochastic Trading Behavior Model for Financial Markets based on Residual Blocks or Transformers,” in *Proceedings of the 35th Annual Conference of the Japanese Society for Artificial Intelligence*, pp. 2N1-IS-2a-03, 2021.

- [86] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.
- [87] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” *arXiv*, no. 1784, pp. 1–7, 2014.
- [88] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” *arXiv*, 2015.
- [89] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least Squares Generative Adversarial Networks,” in *Proceedings of IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017.
- [90] S. Nowozin, B. Cseke, and R. Tomioka, “F-GAN: Training Generative Neural Samplers using Variational Divergence Minimization,” in *Proceedings of 30th International Conference on Neural Information Processing Systems*, pp. 271–279, 2016.
- [91] E. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks,” *Advances in Neural Information Processing Systems*, vol. 28, pp. 1486–1494, 2015.
- [92] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, “Autoencoding beyond Pixels using a Learned Similarity Metric,” in *Proceedings of International Conference on Machine Learning*, pp. 1558–1566, 2016.
- [93] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image Translation with Conditional Adversarial Networks,” in *Proceedings of IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.
- [94] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, “Stock Market Prediction Based on Generative Adversarial Network,” *Procedia Computer Science*, vol. 147, pp. 400–406, 2019.
- [95] C. Chu, A. Zhmoginov, and M. Sandler, “CycleGAN, a Master of Steganography,” *arXiv*, 2017.
- [96] T. Karras, S. Laine, and T. Aila, “A Style-based Generator Architecture for Generative Adversarial Networks,” in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.
- [97] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” in *Proceedings of 6th International Conference on Learning Representations*, 2018.

- [98] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial Feature Learning,” *arXiv*, 2016.
- [99] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery,” *Information Processing in Medical Imaging*, vol. 10265, pp. 146–147, 2017.
- [100] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, “Efficient GAN-Based Anomaly Detection,” *arXiv*, 2018.
- [101] D. Li, D. Chen, J. Goh, and S.-k. Ng, “Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series,” *arXiv*, 2018.
- [102] M. Arjovsky and L. Bottou, “Towards Principled Methods for Training Generative Adversarial Networks,” *arXiv*, 2017.
- [103] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” 2017.
- [104] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, “Improved Training of Wasserstein GANs Montreal Institute for Learning Algorithms,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 5767–5777, 2017.
- [105] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral Normalization for Generative Adversarial Networks,” in *Proceedings of 6th International Conference on Learning Representations*, 2018.
- [106] J. Li, X. Wang, Y. Lin, A. Sinha, and M. Wellman, “Generating Realistic Stock Market Order Streams,” *AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 727–734, 2020.
- [107] Y. Naritomi and T. Adachi, “Data Augmentation of High Frequency Financial Data Using Generative Adversarial Network,” in *Proceedings of 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pp. 641–648, IEEE, 2020.
- [108] A. Coletta, M. Prata, M. Conti, E. Mercanti, N. Bartolini, A. Moulin, S. Vyretrenko, and T. Balch, “Towards Realistic Market Simulations: a Generative Adversarial Networks Approach,” in *Proceedings of the Second ACM International Conference on AI in Finance*, pp. 1–9, 2021.
- [109] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao, “Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets,” *Mathematical Problems in Engineering*, vol. 2018, 2018.

- [110] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [111] R. S. Sutton, “Learning to Predict by the Methods of Temporal Differences,” *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.
- [112] G. Tesauro, “Temporal Difference Learning and TD-Gammon,” *Communications of the ACM*, vol. 38, no. 3, 1995.
- [113] G. A. Rummery and M. Niranjan, *On-line Q-learning using Connectionist Systems*. University of Cambridge, Department of Engineering Cambridge, England, 1994.
- [114] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level Control through Deep Reinforcement Learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [115] M. G. Bellemare, J. Veness, and M. Bowling, “The Arcade Learning Environment: An Evaluation Platform for General Agents,” *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.
- [116] H. Van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning,” in *Proceedings of 30th AAAI Conference on Artificial Intelligence*, pp. 2094–2100, 2016.
- [117] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, “Dueling Network Architectures for Deep Reinforcement Learning,” in *Proceedings of 33rd International Conference on Machine Learning*, vol. 4, pp. 2939–2947, 2016.
- [118] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, “Noisy Networks for Exploration,” *arXiv*, 2017.
- [119] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.
- [120] OpenAI, “OpenAI Baselines: ACKTR & A2C,” 2017.
<https://openai.com/blog/baselines-acktr-a2c/>.
- [121] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, “Rainbow: Combining Improvements in Deep Reinforcement Learning,” in *Proceedings of 32nd AAAI Conference on Artificial Intelligence*, pp. 3215–3222, 2018.

- [122] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, “Distributed Prioritized Experience Replay,” *arXiv*, 2018.
- [123] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [124] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, “Recurrent Experience Replay in Distributed Reinforcement Learning,” in *Proceedings of International Conference on Learning Representations*, pp. 1–15, 2019.
- [125] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, “A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
- [126] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis, “Mastering the Game of Go without Human Knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
- [127] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous Control with Deep Reinforcement Learning,” *International Conference on Learning Representations*, ICLR, 2015.
- [128] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft Actor-Critic Algorithms and Applications,” *arXiv*, 2018.
- [129] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor,” *Proceedings of 35th International Conference on Machine Learning*, vol. 5, pp. 2976–2989, 2018.
- [130] I. Maeda, D. deGraw, M. Kitano, H. Matsushima, H. Sakaji, K. Izumi, and A. Kato, “Deep Reinforcement Learning in Agent Based Financial Market Simulation,” *Journal of Risk and Financial Management*, vol. 13, no. 4, 2020.
- [131] H. Buehler, L. Gonon, J. Teichmann, and B. Wood, “Deep hedging,” *Quantitative Finance*, vol. 19, no. 8, pp. 1271–1291, 2019.
- [132] I. H. Witten, “An Adaptive Optimal Controller for Discrete-time Markov Environments,” *Information and Control*, vol. 34, no. 4, pp. 286–295, 1977.

- [133] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.
- [134] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy Gradient Methods for Reinforcement Learning with Function Approximation,” *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.
- [135] R. J. Williams, “Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning,” *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.
- [136] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 1, pp. 448–456, 2015.
- [137] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization,” *arXiv*, 2016.
- [138] F. Yu and V. Koltun, “Multi-scale Context Aggregation by Dilated Convolutions,” *arXiv*, 2015.
- [139] J. H. Lim and J. C. Ye, “Geometric GAN,” *arXiv*, 2017.
- [140] S. Kullback and R. A. Leibler, “On Information and Sufficiency,” *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [141] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs Trained by a Two Time-scale Update Rule Converge to a Local Nash Equilibrium,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [142] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved Techniques for Training GANs,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [143] B. Zoph and Q. V. Le, “Neural Architecture Search with Reinforcement Learning,” *arXiv*, 2016.
- [144] E. F. Fama, “Efficient Capital Markets: A Review of theory and Empirical Work,” *The Journal of Finance*, vol. 25, no. 2, p. 383, 1970.
- [145] C. Chiarella and G. Iori, “A Simulation Analysis of the Microstructure of Double Auction Markets,” *Quantitative Finance*, vol. 2, no. 5, pp. 346–353, 2002.

- [146] M. Hirano, R. Wakasugi, and K. Izumi, “Analysis of Demand Response Scenarios by Industrial Consumers Using Artificial Electric Power Market Simulations,” *Proceedings of 12th International Congress on Advanced Applied Informatics*, pp. 547–554, 2022.
- [147] M. Hirano, R. Wakasugi, and K. Izumi, “Analysis of Carbon Neutrality Scenarios of Industrial Consumers Using Electric Power Market Simulations,” in *Proceedings of the 24th International Conference on Principles and Practice of Multi-Agent Systems*, pp. 90–105, Springer, Cham, 2023.
- [148] S. Kohda and K. Yoshida, “Analysis of high-frequency trading from the viewpoint of tick distance and Execution [in Japanese],” in *Proceedings of the 22nd meeting of Special Interest Group on Financial Informatics of Japanese Society for Artificial Intelligence*, 2019.
- [149] H. Matsushima, T. Uchitane, J. Tsuji, T. Yamashita, N. Ito, and I. Noda, “Applying Design of Experiment based Significant Parameter Search and Reducing Number of Experiment to Analysis of Evacuation Simulation,” *Transactions of the Japanese Society for Artificial Intelligence*, vol. 31, no. 6, pp. 1–9, 2016.
- [150] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, “Emergent Tool Use From Multi-Agent Autocurricula,” in *Proceedings of the International Conference on Learning Representations 2020*, 2020.
- [151] G. E. Uhlenbeck and L. S. Ornstein, “On the Theory of the Brownian Motion,” *Physical Review*, vol. 36, no. 5, p. 823, 1930.
- [152] P. Wawrzyński and A. K. Tanwani, “Autonomous reinforcement learning with experience replay,” *Neural Networks*, vol. 41, pp. 156–167, 2013.
- [153] J. Frankle and M. Carbin, “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks,” *Proceedings of 7th International Conference on Learning Representations*, 2018.
- [154] F. Corsi, *Measuring and Modelling Realized Volatility: from Tick-by-tick to Long Memory*. PhD thesis, Università Della Svizzera Italiana, 2005.

謝辞

本博士論文は、筆者が東京大学大学院工学系研究科システム創成学専攻和泉研究室在籍中に行った研究の一部をまとめたものです。

本研究の一部は、科研費 21J20074 の助成を受けたものです。

また、本研究は、日本取引所グループのご支援を受け、実施されました。本研究に使用したデータの提供を頂きました。この場を借りて心より御礼を申し上げます。

本研究に際し、ご指導を頂きました、指導教員の和泉潔教授に感謝申し上げます。また、一部の研究において、共著者として研究のアドバイスをいただきました、坂地泰紀特任講師にも御礼申し上げます。

博士論文の審査にあたり副査を引き受けてくださった、北海道大学の野田五十樹教授、東京大学の藤井秀樹准教授、合田隆准教授、石田隆講師には、有益な数多のご指摘を頂きましたことを、この場を借りて御礼申し上げます。特に、藤井秀樹准教授には、学部時代より、さまざまなことを教わり、研究の「いろは」である、マシーンの組み立てなどを教わったことは今でも忘れられません。本当にありがとうございました。

研究室においても、非常に多くの皆様にお世話になりました。特に、私の研究生活の大半において、研究室の秘書としてお世話になりました、山本由香さんにも心より御礼申し上げます。

そして、この博士課程を無機質なものに変えた COVID-19 も忘れられないものとなりました。多大なる自宅での研究時間を生み出したという点では、この博士論文にも寄与したことでしょう。また、COVID-19 を通して、マルチエージェントシミュレーションの有効性が社会的に知られたことも、本研究の追い風になったことは間違いないです。しかしながら、社会を分断し、研究者コミュニティにも多大なる影響を及ぼしたという点では、被害が大きく、再び同様の事態が起きないことを願います。

この博士論文を完成させるまでの軌跡をさかのぼると、8年前に至ります。8年前の私が学部1年生の時に取った授業が和泉先生との初めての出会いでありました。学部1年生ながら、社会シミュレーションとはどういうものであるのか、ということを学び、生意気ながらもシミュレーションを作成し、和泉先生とディスカッションをしたことは、今でも忘れられません。そして、構造計画研究所の MAS コンペティションに参加しました。今、振り返れば、社会シミュレーションに関連するテーマで博士論文を書くに至った原点はそこにあったと思います。そして、学部3年生から、今の和泉研でお世話になり始め、本格的な研究をしました。学部3年生の時に初めて作成した研究論文では、人工知能学会全国大会の学生奨励賞を獲得しましたが、これもまた、人工市場を用いた研究でした。その

後、テキストマイニングの研究などにも従事するようになり、より幅広く技術を学んだ結果、データマイニングとシミュレーションを融合させる研究をするようになりました。私がこれまでやってきた研究は、どれも、現在の博士論文の技術的な礎となっており、決して切っても切り離せない経験でした。これらの非常に貴重な研究経験を、非常に長きにわたって、この和泉研究室で経験できたことは、とてもありがたいことだったと思います。そして、この機会を常に与え続けてくださった、和泉潔教授には、大変感謝しております。

また、私の大学生活は、常に刺激にあふれたものがありました。学部時代、東京大学の支援で、3回にわたる短期留学に行き、修士時代には、海外武者修行プロジェクトとして、大学院工学研究科の支援を受け University College London (UCL) における在外研究を行いました。これらのおかげで、国際学会などを通じて、海外の研究者と臆することなくコミュニケーションをとることをできるようになりました。私の研究生活がさらに有意義なものとなりました。また、卓越大学院 (WINGS-CFS) の履修生としても、非常に多くの学びの場をいただきました。他にも、在学中には、東京大学内の情報基盤センター、本部、教養学部、工学系研究科など、様々な場所で様々な経験をする機会を得るとともに、多くの方々にお世話になりました。さらに、国際学会等でお会いした非常に多くのアカデミアの関係者には、様々な刺激を頂くとともに、研究コミュニティとして、暖かく私を受け入れてくださいました。これらの経験は、必ずしもこの博士論文に直接関連するものではありませんが、多大なる影響を受けていることには間違ひありません。そのような多くの学びの機会を与えてくださった、様々な機関・部署の皆様に、この場を借りて、御礼申し上げます。

そして、私が7年にわたって所属した、システム創成、つまり、工学部システム創成学科と大学院工学系研究科システム創成学専攻は、非常にユニークな環境でした。非常に多くの分野の非常に多くの先生が所属しており、また、学生もユニークです。決して自分の分野にとらわれることなく、様々なことを学ぶことの重要性を強く学ばされました。そして、そういった様々な分野の話を聞いて、インスピレーションを受けられるような面白い環境だったと思います。

また、この博士論文を完成させるに至るまで、様々な友人達にも支えられてきました。研究で躊躇いたとき、論文が国際会議に落ち続けて、もう立ち直れなくなるのではないかと思ったときもありました。そんな時でも、話を聞いて、飲み相手になってくれて、心の支えになってくれた友人達には感謝してもしきれません。

この博士論文は、学位取得という意味では重要なのですが、人生においては、通り道の一つの副産物でしかありません。これから広がるであろう様々な可能性やチャンスをさらに生かし、公私ともにさらに頑張ろうという決意を新たにして、これから日々を過ごしたいと思います。

そして最後に、いつも支えてくれた家族に最大の感謝の意を表し、本論文の謝辞とします。

2023年8月
平野 正徳