日本大学経済学部経済科学研究所研究会

【第227回】
2025年7月31日

2023～2024年度共同研究B成果報告

「経済システムに関する数理モデルを用いた解析」

〈講演者〉
日本大学経済学部教授　栗　野　俊　一

日本大学経済学部教授　三　井　秀　俊

日本大学経済学部専任講師　戸　塚　英　臣

栗野　本日は経済学研究所の共同研究の発表会に参加していただいてどうもありがとうございます．われわれは『経済システムに関する数理モデルを用いた解析』と言うタイトルで 2023 年度，共同研究を採用していただきまして，今日はその発表会をさせていただきたいと思います．まず最初に紹介なんですけれど，今あいさつしてる僕が，研究代表の栗野です．どうかよろしくお願いします．さて，この 3 人，理系の人間らしく，そのためか，選んだテーマは，数理システムを用いて経済現象に対して何らかの結果を得ようと言う内容になっています．三人三様の内容で，今日は三つの話題を進めていきたいと思います．本日は，録画をすると言うことなので，このままずっと録画をしますので，大変，申し訳ないんですけど質疑応答に関しては最後にまとめてしていただくと言う形にしたいと思います．どうかよろしくお願いします．

それでは早速ですけれど最初は，戸塚先生に発表を始めていただきたいと思います．じゃあ変わります．ちょっと準備に時間かかるかと思いますけれど，少々お待ちください．

戸塚　これで大丈夫でしょうか．それでは最初に話をして．『マルチフラクタル解析を用いたボラティリティのラフ性の実証分析』と言うタイトルで戸塚が発表いたします．まず時系列データの長期記憶性と言う話なんですが，すごくマニアックな話で大変，申し訳ないですけども，自己相関関数，時間で変動するデータ，主に扱ってるのは株価指数だとか為替だとかそういった日々，変動するようなデータを扱っておりまして，長期記憶性と言うのは，時系列データの過去の影響が長期間にわたって現在もしくは将来に影響を与える性質のことを差します．長期記憶性は，自己相関が遅く減衰することで定量化されて，下にお示しした三つのグラフは左側から長期記憶過程といって，自己相関が非常に裾が長く残ってる状態を表しています．真ん中が，標準ブラウン運動といわれているもので，こちらは相関がないと言うか，ほぼ自己相関が 0 になっておりまして，一番右側が反相関過程と言う形で最初に同じ時刻なので相関係数が 1 なんですが，ちょっとずれるとマイナスになってプラスになってマイナスになってみたいな反相関的な過程を示してます．下に示した H と言う値なんですが，これが長期記憶過程を特徴付ける Hurst 指数だとかホルダー指数だとかっていわれるものでして，一般的には 0.5 よりも大きい値を取ったときにそのデータは長期記憶過程と呼ばれていて，0.5 を取ったときが標準ブラウン，0.5 よりも小さくなると短期反相関過程と言うふうに分類されるパラメータと言うか指数になります．

ボラティリティのラフ性と言うのは何かと言うと，単純に言うとこの一番右側の短期反相関過程であると言うのがこのラフ性と言うことなんですが，もうちょっとまとめると，ラフ性と言うのは，今，考えてるボラティリティと言うのは，株価市場における価格変動の尺度と言うふうに捉えて，ボラティリティいろいろ定義あると思うんですが，価格の変動もしくは高頻度データから算出されるリアライズボラティリティもしくは日経平均ボラティリティインデックスやアメリカの S&P500 のボラティリティである CBOE VIX などが変動だと思っていただくと，これが単なるランダムウォークやガウス過程のように見えるんですが，ちゃんと分析をしてみるとより複雑な動的構造を持つと言うことが分かってまいりました．

先ほどのグラフで言うと，通常この Hurst 指数が 0.5 って言うのがモデル化された場合の値なんですが，どうもその値じゃないって言うことがいろいろ検証されてきて報告されてきているのがこのラフ性と言うお話でして，金融市場における価格の変動が，単純な確率過程では表現できないことを示唆しており，より深い構造的な相関や長期記憶性と言うかどちらかと言うと短期記憶性になるかと思うんですが，それを示してると言うようなお話になります．

先行研究とすると，もうタイトルがその名のとおりなんですが，Volatility is Rough といって論文が Quantitative Finance のほうに 2018 年に報告されていまして，経験的モーメント関数といわれるようなこのシグマがボラティリティと言うか標準偏差だと思っていただくと当然その対数をとったものの差からモーメントと言う関数を定義して，これは実際リアライズボラティリティといわれる高頻度データから算出されてるデータを使ってるんですけども，さっきホルダー指数って書いてあったんですが，Hurst 指数とほぼ定義は一緒だと思うんです．この関係式で調べると，大体

Hurst 指数がホルダー指数が 0.1 ぐらいあるので，短期記憶性と言うことでラフ性を示していると言う話です．

ちょっと上にお示ししたのが，これは論文のグラフでして傾きがほぼ一定になってるんですが，これは別に同じこの計算をしてみたら，いやそんなことで傾き変わるよなと言うふうにお示ししたんですけども，そんなに単純じゃないのかなって言う，後でまた具体的にお示しします．それをこの Volatility is Rough と言う論文が報告された後に，本当にそうなって言うことでその下の論文のタイトルです．Is Volatility Rough? って言うのがこれは日本の研究者の方がたが報告されてるんですが，ちょっと分析方法が違っていて，局所 Whittle 法と言う長期記憶性を調べるための統計的な方法があって，それを使って調べてみた値なんです．すいません，ちょっとちっちゃくて申し訳ないんですが，そこから，これ実データで S&P 500 と日経平均株価されていて，これもやっぱり 0.5 よりも小さいと言うことでやっぱりラフなんだろうなって言うことが報告されてます．

もう一つ方法論として，今回そのマルチフラクタルと言う話なんですが，これはラフボラティリティのビットコインと言うタイトルでして，ビットコインデータのリアライズド・ボラティリティがラフ性を示すと言うか，正確にはマルチフラクタル性を示すと言う結果が左側のグラフでして．これどう見るかって言うと，q という指数を表すパラメータを振ってあげると，Hurst 指数が一定の値でなくて q と言う先ほどの式で言うとこの肩の値ですね．q の値を変えてあげると，Hurst 指数が非線形な分配を示すって言うことが示されました．

これはビットコインが左側，S&P500 が右側なんですけども，どちらも非線形な振る舞いをしていて，上のグラフと下のグラフは何かと言うと，下のグラフが時系列データの元データです．上のグラフはそれをランダムに入れ替えて，何度もランダムに入れ替えてデータの並び順を無作為に変えてあげて何度もしてあげると，データが関連性があるからフラクタル性を示すんじゃないかって言う話だったので，全部シャッフルしてあげて何度もしてあげればそのランダム性が消えるので，モノフラクタルと言う直線が出るんじゃないかといったらそうでもなくて，こんなやっぱりフラクタル性を示すと言うグラフです．

先ほどモノフラクタルとマルチフラクタルと言うお話をしたんですが，右側のグラフを見ていただくと，この赤い直線になるものがモノフラクタルといって単一のスケーリング指数，Hurst 指数によって特徴付けられるもの．一方，曲線描くような場合が，これがマルチフラクタルといって，単一ではない q と言うスケーリング指数，次数によってフラクタル性と言うかスケーリング指数が変わってくるって言うような性質を示すと言うことが分かっていて，ここは何を言ってるかと言うと，次のとこでもうちょっと詳しく説明しますが，そもそも何を見てるかってお話なんですが，この Hurst 指数とか，今，左側にお示ししたこの関数は，三つの点で連続だけど微分不可な点があります．それぞれ指数が左側から 0.3 乗，0.6 乗，1.2 乗と言う指数を持っていて，ちょうどこれがデータのとがり具合を表してると思ってください．このデータのとがり具合を見てるのが Hurst 指数になります．

そうすると，0.3 乗，0.6 乗，1.2 乗ですね，これ分析してあげると，それぞれ傾きがこう言う傾きとして評価することができます．そうすると，これまでのモノフラクタルとかフラクタルの解析だと，この辺り全部，平均化されてしまうので，本来であれば時系列データの対数収益率とかを見るとデータがぎざぎざとなるんですけど，そのぎざぎざのとがり具合がちょうどこの関数上の型の次数だと思っていただくと，本来だったらいろんな次数をとっているはずなんですけども，それが平均化されて評価されてしまうって言う問題がある．

じゃあマルチフラクタルの何がいいかって言うと，小さい次数と大きい次数が重み付けされるので，すいません，ちょっとあちこちで，q の重み付けされるので，q の値がマイナスをとったときには小さい次数がよりフォーカスされていって，q の値が大きくなってプラスになると，今度は大きい次数がフォーカスされていきます．ですので，モノフラクタルでものを見てるような Volatility is Rough とか Is Volatility Rough? と言う方法で分析してるやつだと，見えないものがマルチフラクタル解析だと見えてると言う利点があります．

本研究の目的なんですが，マルチフラクタル解析を用いてボラティリティのラフ性を明確にすると言う話なんですけど，まず分析用のプログラムを作りました．なぜかと言うとこのマルチフラクタル分析には大きく分けて二つ方法があって，一つ は Multifractal Detrended Fluctuation Analysis と言う，一般的には MF-DFA と言う方法があって，これは R だとか Python でライブラリが既にあります．ただ，コンピューターでやってくと，他人さまが作ったライブラリはあまり信用ができないので，それを使って分析って言うのが嫌だったので，ちゃんと自分で作って精度検証したいなと思ったので，これは作っています．

もう一つ WTMM と言う方法なんですが，これはウェーブレット変換を用いマルチフラクタル性を分析する方法なんですけども，これには満足なライブラリがありません．R や Python でも調べてみたんですが．ですので，自分で実装してみて作ってみました．やりたいのはまずボラティリティは本当にラフ性を示すのかって言うところ，先行研究をちゃんと確認できるのかって言うことと，ボラティリティがモノフラクタルなのかマルチフラクタルなのか．あと，このラフ性の原因ですね．何がマルチフラクタル性を示してるのかって言う話ができればいいなと思って，まず最初にプログラムを作ってちゃんと先行研究と整合が取れるのかって言うところをやってみました．

二つ方法があって，一つは MF-DFA と言うのが，これは後で細かく説明しますが，これの強みは非定常な時系列データに適用が可能です．局所トレンドといって，部分的にトレンドを除去することができるので，一般的に時系列データだと定常性を担保しなきゃいけないんですが，定常性を担保しなくても分析できる方法になってます．

もう一つはウェーブレット変換を用いた方法なんですが，こっちはウェーブレットの性質であるモーメント消去性と言う非常にウェーブレット変換で大事な性質なんですが，それを使ってあげると同じようにトレンドが除去できると言う話があって，大体この二つが大きくマルチフラクタル解析をするときに使われてる方法になります．

この辺りは一応，手順なんですが，ウェーブレット変換を行って一番大き言うェーブレット変換の係数を求めてあげて，その係数に基づいてパスを決めてあげて，そのパス上で確率変数と言う速度を求めてその速度の重み付けをした分配関数を求めて，その分配関数がスケーリング速を満たすので，そこからスケーリング速で指数を求めるって言う流れになります．

MF-DFA の場合はこっちはちょっと厄介で，各データ区間を細かく分けてあげて，それぞれの区間ごとにセグメント分割した後にそれぞれのセグメントごとにトレンドといわれる多項式で傾きを削除してあげて，揺らぎを求めてその揺らぎに重みを付けた平均化してあげて q 次のモーメント求めて q 次のモーメントがスケール則を満たすと言う過程の下で同じようにこのスケーリング指数を求めていくと言う話です．ですので，どちらも基本的にはスケーリング則なんで，何とかの何とか乗と言う形になっていて，これを求める形になってます．

二つの方法の比較なんですけども，MF-DFA のほうは，累積偏差の局所トレンド除去って言う方法を取ってるので，こう言う方法を取ってます．WTMM のほうは，基本的にはウェーブレット変換を用いてウェーブレット変換の係数が一番大きいところをトレースしていって，その傾きを求めているのが WTMM になります．トレンド除去の方法が，ちょっと違うんですが同じようにトレンド除去してあげて，ノイズ耐性と言うかそれぞれ特徴があるんですが，こっちのほうがノイズには強いです，MF-DFA のほうが．

解像度にすると，恐らくウェーブレット変換を使ったほうが解像度が高いかと思います．MF-DFA のほうがいろんなライブラリが整理されてるのは，比較的，実装が容易なんですね．そんな面倒くさくないです．一方，ウェーブレット変換の場合はちょっと面倒くさいところがあるので，敬遠されているのかなと思うんですが，どちらにも特徴があるので両方で調べてみるのがいいのかなと言うふうに考えました．

実データなんですけども，Realized Volatility といわれている高頻度データから算出されるボラティリティの推定値について調べました．これは先行研究で行われている Volatility is Rough とかイフボラティリティラフとかいわれてるものでも同じデータが使われたので，比較のために同じデータを使いました．今は提供されていないんですが，

オックスフォードのリアライズライブラリと言うところから算出したデータで，一応ネットを調べるといまだに使えるかと思います．元サイトはもう閉じてしまっていろいろなところが魚拓をいろいろ出してくれてます．データ区間は大体 20 年間ぐらいのデータをやってます．日経平均株価，S&P500，ダウ・ジョーンズあと各国ごとの株価指数のデータがあるので，サンプルにはいいかなと思います．

これと比較するために，実際のボラティリティ指数として日経平均 VI と CBOE VIX．これは CBOE VIX は，S&P500 に対応したボラティリティインデックスなんです．別名，恐怖指数といって株価が下がると言うときには上がる指数になっています．実際，本当はやんなきゃいけなかったのがちゃんとテストデータを使って精度検証しなきゃいけないんですけど，この辺りはテストデータがそもそも作れな言うまく．マルチフラクタル性を示すようなデータを作るようなプロセスがない．なので，どう言うふうに作ったらマルチフラクタル性を示すのかって言うのが正直よく分からないんで，今もう実データでやってみました．

こちらが日経 225 で左側が MF-DFA，右側が WTMM でやった結果なんですが，同じデータなんですけどもこれぐらい絶対値が変わります．厄介なことに，ちょうどこの真ん中が 0.5 なので，0.5 よりも大きいか小さいかで長期記憶性か短期記憶性かって言うことは分かれるんですけども，MF-DFA だとラフどの次数でもラフだって言うふうにいえるんですが，ウェーブレット変換を用いると q がマイナスの値を取る，つまり非常に小さい変数，言い方悪いですね，速度が小さい値の場合，この辺りは長期記憶性を示す．逆に速度が大きいところは，短期記憶性を示す，ラフ性を示すって言うことで，ちょっと結果が違う．同じように右肩下がりにはなってるんですが，絶対値が違うって言う結果になりました．

Volatility is Rough って言う方法の論文の中で使われているモーメント関数を同じように調べてみると，やっぱりフラクタル性を示すのは明確なので，マルチフラクタルって言うことはいえると思うんですけども，なぜ差が出るかって言うところがちょっとよく分からない結果になってます．これが日経 225 で S&P もほぼ同じ結果です．これが日経 225，これが S&P500 なので，ちゃんとデータは別々のデータを使ってるんですが，グラフで言うとほぼ変わりがないって言う結果になってます．

これは株価指数なので実際のボラテリティ株価指数じゃないですね，高頻度データから求められたリアライズボラテリティこれは変動率です．日経平均 VI が，こっちは非常に強いマルチフラクタル性を示していて，MF-DFA と WTMM が結構，似たような形になってるので，違うものがあって似てるものがあるってことは，うまくデータを精査してあげれば差が出てくるのかな，なぜこう言う差が出るのかが調べられるのかなと思って，ちょっと期待はしております．このデータは駄目だけどこのデータは似てるから，じゃあこのデータとこのデータは何が違うのかってことを調べていけば，なんで MF-DFA と WTMM で差が出るのかってことがもうちょっと突き止められるのかなと言うふうにちょっと期待をしております．

ちなみに CBOE VIX 同じような結果になったので，株価指数のデータを使ったリアライズボラティリティのデータと，実際のボラティリティのデータはちょっと特性が違うのかなって言うので，その辺りをうまく比較してあげれば方法論の違いがうまく出てくるのかなと言うふうにちょっと期待はしております．じゃあなんで違いが出るかって話なんだけども，そもそも求めている量が違うんですね．MF-DFA は揺らぎって言う値です．統計用語で言うと標準偏差です．標準偏差を重み付けしてあげて使ってます．

一方こっちは，ウェーブレット変換をしてあげたときのウェーブレット変換係数を使って求めているので，そもそも求めてるのが違う．もう一つ，トレンドの除去の違いと言うのがあって，MF-DFA では加工してフィットをしてあげて局所的なトレンドを除去してます．実はこれは明示的にトレンドを除去してるんですが，ウェーブレット変換の場合には，消失モーメントといって何次のモーメントが 0 になると言う条件を使って，これは明示的ではないんですね．その辺りが違います．もう一つ解像度の取り扱いが，MF-DFA は区間を適当に分けてるので，区間の分け方が違うんですね，ウェーブレット変換．ですので，多分この三つぐらいが差が出ている原因かなと言うふうに今のところちょっと考えています．

と言うことで，そろそろまとめに入りますが，やったことは MF-DFA と WTMM の計算を行う Python のプログラムを開発しました．精度検証については，一応テストデータをいくつか回してみたんですが，やっぱりテストデータでも違いが出ます．日米の株価指数とボラティリティ指数のマルチフラクタル解析を行って，どちらも結果的にはマルチフラクタル性を示す．ただ，絶対値が違うってことが分かったので，特徴とすると株価指数とボラティリティ指数でちょっと特徴が違ったので，データの特性の違いを軸にしてあげて，方法論がなぜ違ってくるのかってことを調べてあげればいいかなと言うふうに考えています．

今後の課題なんですが，結果の違いの原因を追求すると言うことと，MF-DFA はトレンドの除去に局所多項式近似を使ってるんですが，最近，移動平均を使ったほうがいいって言う話があって，マルチフラクタルムービングアベレージなんとかって方法があって，そっちのプログラムを実は作って試験をしてるんですが，それと比較してみるのもいいかなと思ってます．

それから，ウェーブレット変換は，今，連続ウェーブレット変換を使ってるんですが，離散ウェーブレット変換のほうが精度がいいと言う論文もあります．これはもう完全に物理のほうの世界なんですけども．そっちも実装してみたんですが，離散データになってしまうとデータ点が少ないので，あんまり精度が良くないんですね．ですから，一応，拡張は考えているんですけど，拡張してみて MF-DFA を二つ，それからウェーブレット変換は連続と離散で二つぐらいを最終的には報告できればいいかなと言うふうに考えています．以上です．

栗野　ありがとうございました．それでは引き続き次の発表をお願いします．

三井　始めたいと思います．今回は『Hamiltonian Monte Carlo 法による Stochastic Volatility オプションの評価法』と言うテーマで発表させていただきます．最初に研究の背景と目的を説明します．連続時間 SV モデルと離散近似の話をします．次に SV モデルとベイズ推定に関して説明します．連続時間 SV モデルと離散近似，この SV モデルとベイズ推定に関しましては前回の戸塚先生との共同研究と重複していますので，そこは簡単に説明します．4. の SV モデルの拡張と 5. がメインとなります．ベイズ推定による SV オプションの評価法で，今回は戸塚先生にも質問しながら進めてみたいと思います．今回は実証研究ではなく，このような方法でオプションの評価ができるのではないかと言う論文になっています．

まず，研究の背景ですが，先ほど戸塚先生の報告ではありましたように，金融資産の，特にリスク資産の分析ではボラティリティに注目することが非常に多いです．ボラティリティと言うのは収益率の標準偏差のことで，厳密には連続複利収益率の標準偏差になります．ボラティリティって言うのは，先ほどの戸塚先生の説明でもあったように，時間を通じて変動しているわけですか，そう言ういろんな性質がありますが，そのときボラティリティの時系列変動をどのように定式化するかと言うのがファイナンスの分野では一つの大きなテーマになっています．

様々なモデルがありますが，ボラティリティに注目するのはファイナンスの研究の場合には多いのですが，オプションと言うのはデリバティブの一種で，オプションの評価するときに一番重要なのは，ボラティリティの評価をどのように行なうかにあります．オプション価格を導出する変数は 5 つあります．金利，残存期間，原資産価格，権利行使価格とボラティリティになります．最後のボラティリティだけ未知の変数になります．と言うことは，オプションの取引において，トレーダーの人たち，あるいはディーラーの人たちは，結局ボラティリティが今後どのように変動していくかでオプション取引の勝負しているわけです．そのため，オプション価格を定式化し評価するときには，ボラティリティの推定をどのように処理するかが非常に重要な問題になってきます．

基本的に時系列分析でリスク資産の評価を行な

い分析する場合には，Stochastic Volatility モデル（略して SV）を使用する場合が多いです．もう一つは，ARCH 型モデルを使用する場合の 2 つのケースに分かれます．Stochastic Volatility モデルって言うのは，基本的に株価の変動をうまくモデル化するのですが，最初は連続時間で考えているので，それを離散化してまたそれを推定するときにいろんな問題が出てきます．そのため，一般的に実際に金融の現場で使うって言うことはほとんどないですね．研究者しか利用しないような感じです．

ARCH 型モデルは，もともと離散モデルとして定式化していますので，非常に推定が簡単なモデルになっています．ARCH 型モデルって言うのは，ご存じの方もいるかと思いますが，GARCH とか EGARCH とかいろんな種類のモデルがあります．

オプションの価格付けは，今回は時系列を使ったモデルですが，1973 年にブラック・ショールズがオプション価格の公式を導出しています．その功績でノーベル賞を受賞しています．ただ，今もブラック・ショールズモデルでオプション価格を評価してトレーディングをしている人はもうほとんどいないですね．もう過去の遺物です．そのため，基本的にはブラック・ショールズモデルはもう使ってないのですが，いろいろ改良系はありますし，ベンチマークにもなっていますので，いろんな考え方ができるのですが，時系列モデルを使ってオプション評価をしたほうがより現実的なものができるのではないかと思います．

目的は，ここが一番重要で，先ほど言いましたように SV，Stochastic Volatility モデルは，推定するのが難しいと言う問題がありあまり使われてないのですが，今回は以前に戸塚先生と共同研究で Hamiltonian Monte Carlo を使ったそのベイズ推定でパラメータの推定を行なってリスク資産価格の分析を行なう研究をしました．日経平均を使用して．そこで思ったのは，これはオプション価格の評価に応用できると考えました．今回ちょっと荒い話になりますが，このように応用したらオプション評価を Hamiltonian Monte Carlo 法を使ったベイズ推定で評価できることを提案したいと思います．

最初に，離散近似の話を簡単にしておきます．基本的にファイナンスの世界で株価の過程を定式化する場合には，ブラウン運動を用いて定式化します．μ がドリフトで dt が微小変化分になります．σ がここで言う標準偏差で，dz がウィナー過程を示しています．σ は標準偏差ですが，金融の世界では先ほど説明したようにボラティリティと言う場合が多いので，一応この研究では σ 二乗をボラティリティと定義します．σ でも良いのですが．今度はボラティリティの過程も定式化して，ボラティリティの過程もブラウン運動を利用して定式化します．

先ほどの戸塚先生の研究でもありましたが，ボラティリティの過程をどのように仮定するかは，非常に勝負どころでして，オルンシュタイン・ウーレンベック過程のような回帰性を持つモデルを使う場合が多いです．記号の説明は記載しているとおりですが，ただ問題は，これを連続モデルから，どのようにして離散型のモデルにするか，あるいは離散に近似するのかが次の問題です．先ほどの株価の過程とボラティリティの過程を離散モデルに変換します．ここで気を付けないといけないのは，y_t は収益率を表しています．h_t はボラティリティのエクスポネンシャルを取った形になります．

問題は，h_t が潜在変数になるので，これを推定するのが非常に難しいと言う問題が出てきます．今の話は，細かいところは戸塚先生と僕の共同研究や他の先行研究を見て頂きたいです．要は，これまでの話は過去の 0 時点から T 時点までのデータを使って何かしらのパラメータを推定すると言うことです．オプションと言うのは満期日があって満期日までの原資産価格あるいは収益率を計算あるいは推定して，それをまた現在価値に戻してオプション評価をします．ここの部分をどのように処理するかといのが今日のメイン・テーマになります．

ベイズ推定の話は，戸塚先生と私の共同論文を読んで頂ければ，どのように対処するか詳しく書いてありますので，そちらを参照してください．

今，説明したのは簡単な SV モデルの話で，一応，株式市場にはもっと特徴があります．よくいわれる特徴として，株価収益率とボラティリティとの間に非対称な動きの関係があると言われます．それをレバレッジ効果といいますが，そうすると収益率の過程の式とボラティリティの過程の式の誤差同士に相関があると仮定して，そのボラティ

リティと株価収益率の非対称性を捉えるようなモデルを作ります．これも共同研究で戸塚先生としていましてパラメータの推定はできることは分かっています．

もう一つ株価と収益率の性質として，通常，自然科学では正規分布を仮定する場合が多いのですが，リスク資産収益率の分布をよく見ると，尖度の値が高くなります．何を意味しているかと言うと，分布の裾が厚くて平均の周りの頻度が高い分布に従っています．それをファット・テールの分布と言いますが，それを考えます．t 分布を使うとフィットが良いことが分かっていますので，誤差後に t 分布を仮定したモデルをよく使います．これも戸塚先生との共同研究で行なっていまして，パラメータの推定は可能です．

本題はここからです．要は 0 時点から t 時点までの区間のデータを使ってモデルのパラメータの推定ができます．今度はそのパラメータを使って満期までの原資産収益率を求めます．通常，今までの先行研究だと，この時点を何かしらのモデルでパラメータを推定して，現在時点から将来に向かってなんらかの計算する場合には，モンテカルロ・シミュレーションや別のシミュレーション方法を使用することが多いです．それだといろんな誤差が出てしまい，1 度パラメータを推定してまた別の方法で将来の株価を推定すると言う，何て言えばいいですかね，パラメータを推定している区間の方法とその後の方法が異なると言うのは整合性がとれていないと思います．また，誤差がかなり生じると考えますので，ベイズ推定で行なうと言うのは，その問題を少し解消できると思います．どうでしょうか，戸塚先生．

戸塚　データは t 時点までデータがあるから，その t 時点より前のところまでのデータでパラメータ推定して，そこから 1 日ごとに進めていって，最終的に t 時点までいって，そのときにどれぐらいのスケールがあると言う評価はされています．

三井　実際，ベイズ推定では，モデルが一つで一気にここまでいけませんか？

戸塚　できるんですけど，結局やることは先生がおっしゃるように，同じ SV モデルを使ってシグマが決まっているので，あとは単純に正規分布に従う，もしくは t 分布に従うときのパラメータをそのまま使って，そこで乱数を発生させて，どんどん次を推定していき，それを何度も行なえば，期待値として T 時点での収益率を推定することができます．

戸塚　T 時点までが何日か分からない．大体，半年ぐらい？

三井　短いのは 1 カ月とかです．
戸塚　1 カ月ぐらいだったら，恐らく，多分，ランダム・ウォークじゃないけど，ある程度，推定して T 時点での分布を調べて推定はできます．

三井　そうですね．よかったです．オプションと言うのは，コール･オプションとプット･オプションに分かれる．満期日でのその原資産価格から，権利行使価格を引いて，0 か正でどちらか大きいほうの値を取ります．その平均を取ったものがオプション価格になります．問題は無リスク資産の金利で現在価値に割り引いているのですが，そこの処理が，実はそれで良いのかと言う問題があって．本当は，投資家のリスク中立性を仮定しなければならず，何かしらの測度変換をしてリスク中立の世界にしなければなりません．ただ，今まで金利が低かったので，別にそう言うことは気にしなくてよかったのですが，今は金利が上がってきたので，単純に無リスク資産の金利で現在価値に割り引いて良いのかと言う問題が別途，出てきます．

ただ，オプションの評価するときに，きつい仮定になるのですが，投資家はリスク中立の投資家を仮定すれば，ただ単に，満期日の原資産価格を現在価値に割り引いて使用すると言うことで良いと考えます．ただ，これは今後の課題になります．

結局，0 時点から t 時点まで，過去データからベイズ推定して，そのまま直接，満期日までの原資産価格も推定します．それを割引現在価値にして，現在のオプション価格の推定ができると言うことです．今回の研究は，Hamiltonian Monte Carlo 使ってベイズ推定を利用すると，一度でオプション価格が導出することができると言うことです．尻切れトンボみたいになりましたが，これで終わりにしたいと思います．

栗野　ありがとうございました．それでは，最後になりますけれど，僕の発表に移りたいと思います．

栗野　それでは，『極大外平面グラフ最短経路書き換えアルゴリズム』と言う表題で始めたいと思います．どうか，よろしくお願いします．まず，背景になるんですけれど，グラフと言うのがあって，点の集合と，その間を結ぶ辺からモデル化されるような図形があるわけですね．例えばこんなグラフ（図 1）があると思ってください．この丸いのが点で，これらが辺で接続されているわけですが，その中から，適当な点を二つ選んだ時に，その間を辺だけを通り，できるだけ短い経路でつなげる，最短経路を求めたいって言うような問題を考えましょうと言うわけですね．

ここで改めて，最短経路問題って言うのは何かって言うと，与えられたグラフの 2 点間の最短経路を求めましょうと言うことで，実際に日常的にカーナビを使って経路を探すとか，交通費の計算のための交通機関の経路案内ソフトを使うとかそう言うことをやっていると思いますが，それが正に，この問題を解いている事になると言うわけです．最短経路を使うことによってコストが下がると言うことがあって，これは別に経済現象に限らず，いろんな所で役に立つので，多くの興味を引いたわけです．しかも，これは明らかに分かりやすい問題で，昔から色々な人が解こうとしました．それで，実は結論から言うと，この問題に関しては既に完全に解けています．つまり，これ以上良い方法がないって言う最良の方法が分かっています．

その解法では，問題を二種類に分けて別々に解こうとします．一つ目は，1 カ所を始点としてそこから他の全ての点への最短経路を求める場合で，二つ目は，任意の 2 点間に関して全ての最短経路を求める場合です．なぜ分けるかって言うと，2 点間の最短経路を全て調べる場合，既に調べ終わっている他の始点に関する経路情報があるので，別の所を調べる時に，その情報が使えるので少し楽になる．だから，ぐるぐると始点を変えながら，それを始点として，別々に最短経路求めることを繰り返すってやり方よりは効率よく探せるだろうって言うことですね．

ここで，本来，求めたいのは 2 点間の最短経路なんですが，実際に解く問題はどうなっているかと言うとちょっと大事 (おおごと) になっていて，一つの始点から全部の点への経路を求めようとするわけですよね．それはなんか無駄なことやっているように聞こえるんですけれど，どうにもならないんですよね．つまり，ある経路が最短かどうかって言うのは，別の経路も調べてみて，それと比較して初めて解るんで，特定な 2 点間の距離を最適化しようとした場合であっても，別の候補となる経路も全部調べないといけないから，結局，全部の経路を調べることになっちゃうってそう言う仕組みになっていると言うことです．

話，戻しますけど，こう言うグラフがあって始点が要求されたときに，そこからこう言う最短経路を求めるって言う作業をするんですが，これ先ほど言ったようにもう解けちゃっている，だから，改めて研究する価値ないわけですよね．じゃあ僕は何を考えているかって言うと，既にある最短経路を書き換えて新しい経路を作るような問題を考えてみようと思っているわけですね．例えば，初めて行った所で車を運転している状況を想像してみてください．カーナビを使うと最短経路を提示してくれるわけですね．そして，その経路に従って車を進めて行く．その経路通りに進んでいく分には全然問題ないんだけど，うっかり，横道に間違えて入っちゃったなんてことがよくあるわけですよね．その時にどうするかって言うと，もちろん，改めて現在の場所から目的地までの経路を再度，最初から計算しても良いんですけれど，ちょっと始点が変わったぐらいだったら，あまり最短経路が変わるとは思えないわけですよね．そうであれば，既存の計算結果を使って，うまく速くリカバリーできるんじゃないかと考えたわけですね．

そこで，最短経路の書き換え問題を定式化したいと思います．従来の問題は何も知らないところから最短経路を求める．だけど僕はどうしたかって言うと，既に特定な点からの最短経路は求まっているんだけど，その点とはちょっと違う所から出発した場合の計算を速くしようと言う研究を考えたと言うことですね．先ほどの例（図 2）なんですけれど，ここ（点 O）から出発するとこう言

う最短経路になるんですね．それで，始点をちょっと隣の点 S に変えてみる．そうするとどうなるかって言うと，こっち（図 3）に変わるわけですね．さっきと何が変わったかって言うと，今ちょっと四角で示された 3 カ所，書き換わったのは実はこの 3 カ所しか書き換わってない．残りの部分はそのまま同じなんですね．だからやっぱり始点がちょっと変わったぐらいだとすると，全部やり直しをしなくても一部分やり直せばすむだろうと言うことは何となく予想がつくわけです．

じゃあどうやってその変化する部分をうまく見つけて書き換えるかと言うことを考えたわけですよね．一般のグラフの場合こう言うことがあり得るよって言うことをずっとやってて，それが実は僕のドクター論文になっているんですけれど，改めて適応範囲を考えていくと，この書き換えのアイディアが，いつでもうまくいくかって言うとそんなことは全然ないわけですよね．

実際，書き換えが不利な場合がたくさんあって，一番不利なのは点が全部つながってる完全グラフの場合．その場合，何が起きるかって言うと，始点を変えたら，最短経路が全部，変わっちゃう．全部，書き換えなきゃいけないから，元の情報はあってもしょうがない．

それから，書き換え部分が少ない場合であっても，どこを書き換えると良いかがよく分かんない場合ってのあるんですね．例えば次の例（図 4）ですけど，すごく分かりやすいのは直線でつながっている場合．この場合は，始点が一つずれた時に書き換える場所は 1 カ所しかない事が，すぐ分かります．ところが，ちょっと下の 3 と 4 の部分つないだ円環のグラフ（図 5）を考えてしまうと，当然，始点を 0 から 1 に移行した時，さっきと同じ場所を書き換えなきゃいけないんですけど，この追加した部分も書き換えなきゃいけないわけですよね．今回はグラフが小さいからすぐ近くの場所で見つけることができるんですが，これが大きなグラフではすごく遠くなってしまう，ちょっと向こうで書き換えた結果がかなり遠くの場所に影響を及ぼしてしまうって言う現象が起きてしまっているわけで，うまくいかないわけですよね．

この円環と直線の二つの場合を比較してみると，直線の方はものすごく有利で，ものすごく効率良い．でも，円環は全然効率が良くない．それが単に，変な隠れ辺が 1 本あるせいで起きちゃうわけですよね．こう言うのをどうやって考えるかと言うことをやっているわけです．実際にグラフの形状が変わってくると，どう言うふうに変わってくるかって言うのをいろんなグラフで考えるわけです．例えば，さっきも言ったように，完全グラフは全部書き換わっちゃうのでアウトですが，2 次元メッシュぐらいになると，書き換わる部分と書き換わらない部分が半々位でかなり有利になる．

それから，もちろん得意な木になると 1 カ所書き換えれば解決しちゃうのでとても有効なわけですよね．だからやっぱりグラフの形状をよく調べないと，どう言う方法がうまくいくかって言うのがよく分からない．これはアルゴリズムの話じゃなくて，実際にグラフをたくさん作ってどのぐらい書き換えが必要なのかって言うことの統計を取っているって状況なわけです．

じゃあって言うんで，ここから今回の研究になります．書きかえのアルゴリズムが有効になるのは，どんなグラフか考えた結果，どうも極大外平面グラフがどうも有利そうだし，実際に調べたら有利だと言うことが分かり，かつそれを対象に最短経路の書きかえを高速に実行するアルゴリズムを考えましたって言うのが，今回の研究成果になります．

まず，極外平面グラフってどんなのかですけど，これは図（図 6）を見たほうが早いんですけど，要するに多角形があって，その多角形に対して対角線を，互いに交わらないようにできるだけ沢山引いたグラフだと思ってください．これが極大外平面グラフって呼ばれています．名前の由来は，外側に円環グラフがあって，内側だけ極大，つまり，もうこれ以上に追加できないぐらい対角辺が引かれているので極大（これ以上増やせない）なわけです．もちろん，外側にも線を引くこともできるんですけれど，そうしちゃうとうまくいかないわけですよね．

今回の場合は，こう言うふうに内側に対角線があるようなグラフだけを考えましょうと言うわけで，これが極大外平面グラフと言うグラフです．これはいろいろ面白い性質があって，辺の本数が計算できるとか，外に点を追加していくことで，グラフを少しずつ拡大していくことができたりします．まず，辺の本数に関してですが，外側のぐ

るっと一周分の辺の本数がちょうど点の数と同じですね，円環ですから．それから対角線は，中学校のときに習っていると思うんですけれど，点の個数から 3 引けば出てくるので，それで全部の辺の本数が分かるわけです．

あと，面白い性質としては，今，ここに点 5 がある（図 6）んですけど，なかったと思ってください．ないとどうなるかって言うと，これ元の 7 角形から一つ点が減った 6 角形の極大外平面グラフになるわけですね．それに 5 と言う要素を元に戻すと，それはこの辺と辺が加わり，2 辺増えた極大外平面グラフにできる．こうやって極大外平面グラフをどんどん成長させることができますね．成長させることができるので，帰納法を利用して最適性の証明とか，それからアルゴリズムが書けたりすると言う良い点が生まれるわけです．特に大事なのは，一つの極大平面グラフを，二つに分けることができると言う性質で，対角線の所で切って二つのグラフに分けたときに，両方ともこれが極大外平面グラフになると言う大変良い性質があるわけですね．この性質が後で効いてきます．

そう言う良い性質がいくつかあって，それに使ってどうするか，これが一応，今回の結論になるんですけれど，極大外平面グラフにおける書き換え方法の提案，どこから始めても良いんですけど，0 でまず 1 回目の経路作成を行う．この時，メインは書き換えアルゴリズムなんで，1 回目はとにかくダイクストラ法かなんかを使って最適経路を作ります．次に始点を 1 に動かして，そして効率よく書き換えをするって言うことをぐるっと，始点を一周させながら書き換えをする．すると，どんな結果が得られるかと言うと，証明ができているのですが，辺の書き換えの回数が実は完全に分かっちゃいます．書きかえは n 回やることになりますね，点が n 個あるから．それで書き換え回数のオーダーがどうなるかって言うと，これが点の個数と同じオーダーなんです．と言うことは，1 回あたりの平均を計算すると定数オーダーになるので，これはずいぶん速くできる事が予想できる．

ここまではアルゴリズムの高速化の話としてじゃなくて，「書き換え回数がこんだけで済むよね」って言う論理的な予想をやってるわけですが，じゃぁ．アルゴリズムはどうなるかって言うと，基本は，先ほど言ったように始点を決めて，そこから経路を書き換えて行くダイクストラと同じアルゴリズムを適用します．ダイクストラではもちろん，最後までグラフを全部なめて，全ての経路を上書きしちゃうわけですよね．ところが，僕のアルゴリズムでは，途中で停止できます．ここまでやったらもうこの後，やらなくても良いよって言う辺が発見できるので，途中で止められる．途中で止められるから元のアルゴリズムより速くなるって言う理屈です．

あと，それから全ての点に関して，順にそれを始点にして書きかえをやるわけですが，その時には，その始点の変更順序が重要で，必ず円周に沿って始点を変える．そうすると最短経路の変化が小さくなる．反対に，いきなり遠く離れた別の所に始点を変えちゃうと最短経路が，がらっと変わっちゃうんですよね．

円周にそって，ちょこちょこっと少しずつしか始点位置を変えないってやると，最短経路も少しずつしか変化しないし，書き換え回数も少ない．それを一周，回してあげると言うそう言う仕組み．これがグラフに対する基本定理，これは数学的な定理なんですけれど，ポイントは何かって言うと，まず，この極大外平面グラフ（図 7）の対角辺を一つ決めてあげます．そうすると，先ほどちょっと説明したんですが，この辺で，グラフを二つに分けることができるわけですよね．

そのときに一方のグラフの方で始点をこうやって変更してみます．その結果，当然，辺が影響を受けるんですけど，もし，たまたま，この切った線の辺が変更されなかったとします．そうすると何が起きるかって言うと，実はこの後ろの残りの部分の辺も全部，変更されないって言うことが証明できる．具体的な例（図 7）なんですが，ちょうど始点を 0 から 1 に変えたときに，この 0 と 3 を結ぶ辺の向きってのは変わらない．そうすると，実際にやってみると分かるんですが，この後ろのこっち側の辺も実は向き変わらない，これが一つ目の定理．

二つ目の定理は何かって言うと，そう言うふうに 0, 1, 2, …って言うふうに円周にそって始点を変更していくと，当然，辺の向きが少しずつ変わっていくんですけど，その振る舞いを示すのが，次の定理なんです．周の書き換えの計算量と言う

ことで，仮定としては先ほどと同じなんですけれど，周に沿って少しずつ最短経路の書き換えをやる．そうすると何が起きるかって言うと，なんと外周辺，外側のやつに関しては 1 周，全部，回り切ったときに，実はどこで書き換わるか分かんないけど，トータルとして 3 回しか書き換わらない．それから，対角線に関して，これも，どこかで書き換わるかもわからないけど，4 回しか書き換わらない．だから合計でどの辺も高々 4 回しか書き換わらないって言う，そう言うことが証明できてしまう．

直感的にどう言うことを言っているかって言うと，要するに，始点を変えていくって言うことは，逆に，書きかえの対象の辺から見ていくと，始点を変えずに辺の向きを変えてるって言う感覚になる．始点が 1 周するって言うことは，辺の方からみれば，自分が 1 周するわけですよね．辺の取る値って言うのは，辺の両端に点が二点あるんですけど，一方の点が始点に近くて他方の点が遠いか，あるいはその逆か，それとも同じ距離かって言う 3 種類しかなくて．これを＋，－と 0 で表わしている．1 周するって言うことは，0, ＋, 0, －, 0 といった感じで変化するから 4 回しか変化しかない．

それから，この周上の辺の場合は何が起きるかって言うと，始点が切り替わった瞬間に 0 の段階を飛ばしちゃう．4 回のうち 1 回分の変化がスキップされるんで，3 回しか書き換わらないならないって理屈になる．そうやって証明ができちゃう．なので，結局，具体例なんですけれど，この同じ例（図 7）で始点を 0, 1, 2, 3 と言うふうにずらしていくと，外周辺である 2 と 3 を結んだこの辺は，始点が 0 のときには等距離で 0 なんですけど＋に変わって，ここで 2 から 3 への始点の変化の時に，いきなり 0 をスキップして＋から－に変わる．

最後に，－から 0 に戻るので 3 回．それから，この対角線ですね．0 と 3 を結ぶ対角線に関しては，やはり＋から 0 になって，－になって，0 になって，また＋になることで，変化が 4 回しか起きないから，結局，先ほどの定理の例になっていることが直感的に分かるわけです．

と言うことで，二つの基本的な定理が証明できているので，それを使ったアルゴリズムを考えましょうと言うことになります．手続きとしては基本ダイエクストラのアルゴリズムを適用するんだけど，途中の対角線で向きの変更がない辺を見つけたら，そこで終了すると言うアルゴリズムになっている．そうするとどんなことがいえるかって言うと，まず，アルゴリズムの正当性なんですけれど，先ほど言ったように基本定理から，始点から辺の向きを変更して行くんですが，ある対角線の向きが変更されてないことが分かった瞬間に，それ以降はもう変わらないことが分かるので，それ止めても良いって言うことがいえる．

それから，数量的な評価に関して．個々の始点に着目した場合は，たくさん書き換えたりちょびっとしか書き換えなかったりするんですが，1 周回すと，トータルとしては先ほど言ったように，点の個数のオーダーの回数しか書き換わらないことが分かっている．と言うことは，平均して考えれば，実は 1 始点あたりの書きかえ回数は定数オーダーって事が今回の成果になります．かなり速くなりますね．

まとめになっちゃうんですけど，結論としては，最短距離の書き換えアルゴリズムの対象として，極大外平面グラフと言う特殊なグラフに限定をして考えると，従来のアルゴリズムに比べるとオーダーが落ちるぐらいは速くなる．そこはちょっと違うんですね．

それから，完全グラフとか木は，色々な性質を持っているから書き換えに対する振る舞いもすぐ分かるんですけれど，極大外平面グラフは，完全グラフや木に比べて，そんなに明らかじゃないグラフなので，少し広い範囲のグラフに関してそこそこの結果が出てきたので，そこはちょっとうれしいかなと．

ただ，きついんですよね，条件がきつい．例えばどんな条件がきついかって言うと，いろいろきついんですけど，極大と言う条件がきつい．極大の条件はすごく重要で，アルゴリズムを止めるために途中で変化しない辺を見つける必要があるんですけど，この辺がなかったら何が起きるか．つまり極大じゃなかったら何が起きるかって言うんですけど，そうするとそう言う重要な辺が見つかんないわけですね．存在しないから．と言うことは，このアルゴリズム適用できない．だからきちんと止められるようなそう言うものが存在するって言う条件をきちんと追っ掛けなきゃいけないん

ですけど，今回は極大平面グラフで全部，辺がそろっていますって言う極大条件を付けているからうまくいってる．

それからあと，普通，最短経路のアルゴリズムを適用するときに，辺に重みをもたせるのが当たり前なんですけれど，今回は 1 にしちゃってるわけですね．だからこそ，比較対象となる経路の距離の差が整数上で連続に変化していって 0 になる瞬間が必ずできて，そこで，きちっと止められるわけですよね．ところが，重みとして実数値が付いていたりすると距離の差が－からいきなり＋へと 0 を飛び越えちゃうようなことが生じる，そうするとやっぱりきちっと止めることができないから，そこらへんの条件もちょっと考えなきゃいけないとか，いろいろきつい条件があると言うのは相変わらず．でも，今後，少しずつその条件を緩めていこうかなとは思っているわけですね．

今後に関してです．今のところ，なんで今回の極大外平面グラフでうまくいっているかって言うと，やはり先ほど言ったように，「必ず『どっかで止まったら，それ以上，先，行かなくて良いって言う辺』を見つけることができる」と言う点が，すごく強力なわけですよね．本質的にはそれしか使ってない．これが実は何の条件から来ているかって言うと，二つのグラフを分けるときに，その分けた境界の部分が，実は完全グラフになっていれば良さそうってことが大体予想できています．要するに何が起きているかって言うと，境界の部分が完全グラフになっていると，始点を少しずらしたときに生じる経路変化の影響が境界の完全グラフ内での経路変更で吸収できちゃう場合があるわけです．何て言うんですか，境界のグラフが書き換えの影響に対するダンパーになっていて，変化の影響を吸収してくれるんですよね．そこで動かないってことが分かると，その先の所に行かない．でも，そこに抜け穴があると，その影響が通り抜けちゃうわけですよね．なので，そこら辺は何となく一般化できるかなとは思っている．それから，あともう一つ，今回うれしかったのは，1 周，回せるって言うのが結構よかったわけですよね．それは極大平面グラフがハミルトングラフになっているってことなんですけど，一般のグラフには当然そのハミルトン閉路があるって保証は何もないわけですから，それをどうするかって言うような課題も残っている．と言うことで，取りあえず，特殊な例に関して少しだけその状況を拡張できましたと言う報告と，それから今後も，もう少し頑張りましょうと言う内容になっている．以上，こう言うふうになります．最後に．ありがとうございますと言うことで．

（了）

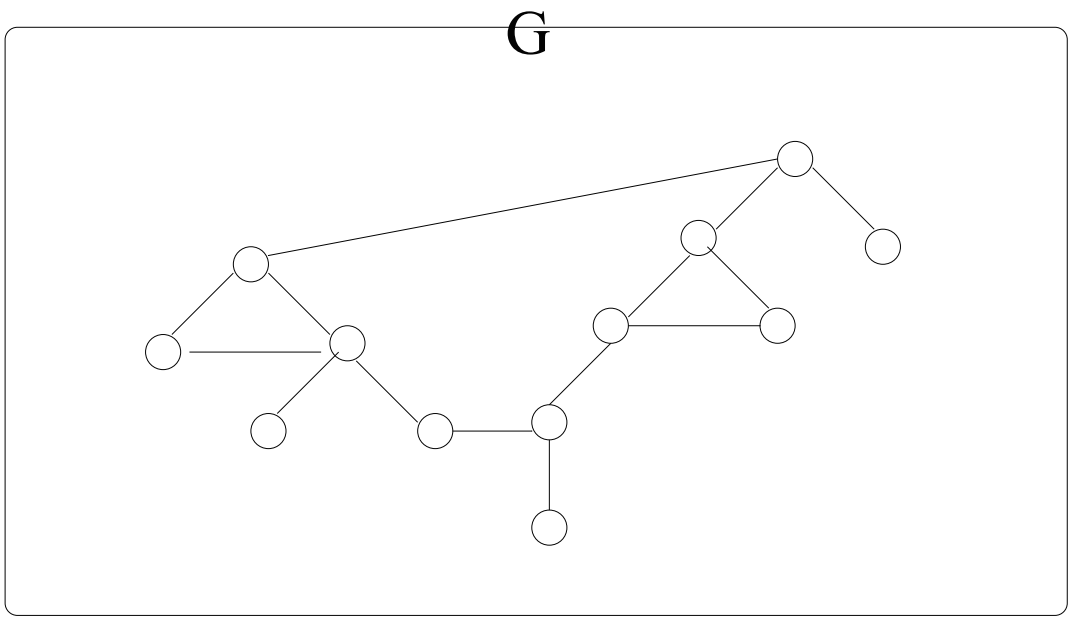

図 1　グラフ $G(V, E)$：問題の対象となるグラフ

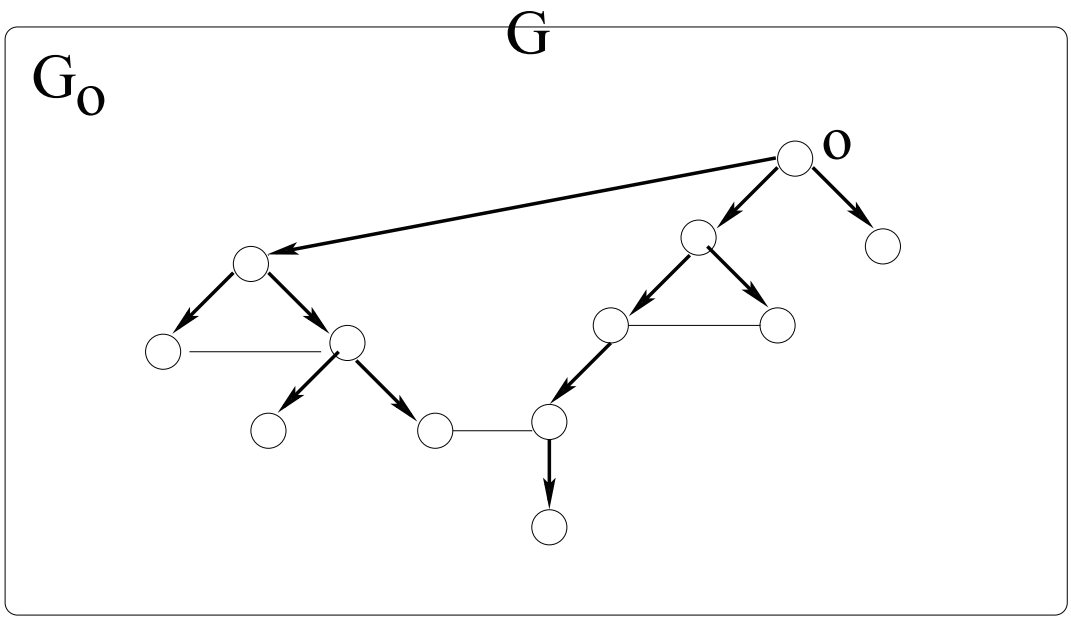

図 2　o を始点とする最短経路網：Go

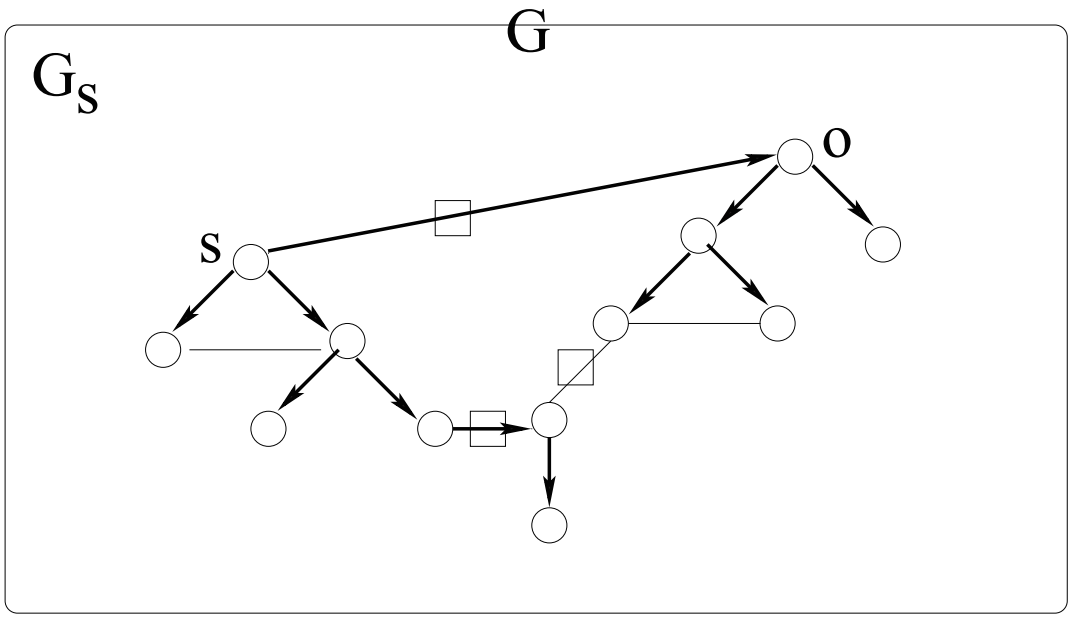

図 3　s を始点とする最短経路網：Gs (Go からの変更点が三箇所)

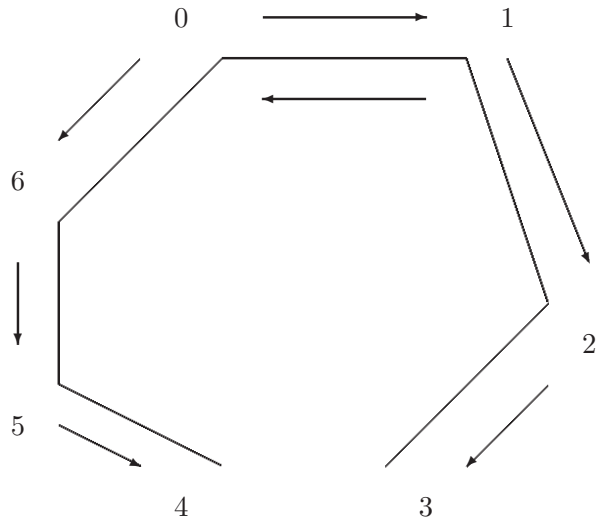

図 4　7 点からなる直線状のグラフ

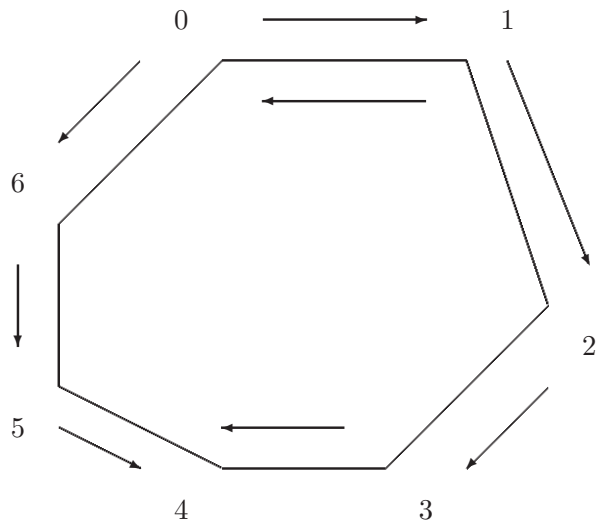

図 5　7 点からなる円環状のグラフ

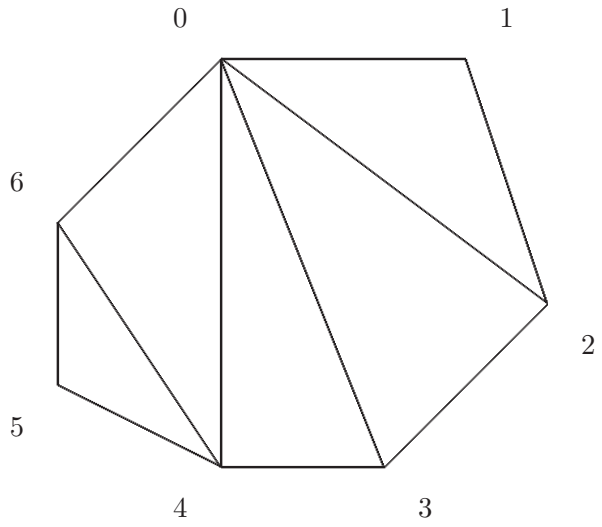

図 6　極大外平面グラフ: 点の個数は 7 点で, 辺の本数は $7 + 4 = 11$ 本

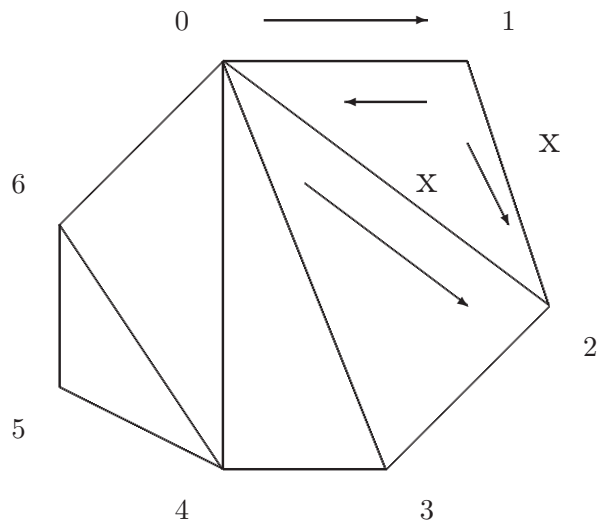

図 7　始点を 0 から 1 に変更 (対角辺 $< 0, 3 >$ が変化しない)