多期間ポートフォリオ最適化問題における モデリング技術と実装(計算)技術の重要性

枇々木　規雄
慶應義塾大学 理工学部
hibiki@ae.keio.ac.jp

田辺　隆人
(株) 数理システム
tanabe@msi.co.jp

第3稿：平成13年 11月 17日
第2稿：平成13年10月24日
初稿：平成13年3月2日

【 連絡先 】 枇々木 規雄
〒223-8522 **横浜市港北区日吉**3-14-1
TEL : 045-566-1635,　FAX : 045-566-1617

【 概要 】

年金基金などの長期的な資金運用を行う投資家にとって、多期間にわたる不確実性を考慮した動的投資政策の決定を「明示的に」モデル化するためには、1期間モデルではなく、多期間モデルを構築する必要がある。本研究では、多期間ポートフォリオ最適化問題において、重要なモデリング技術と実装(計算)技術について議論する。具体的には、問題を実際に解くためのモデルの一つであり、モンテカルロ・シミュレーションによるパスを用いて不確実性を記述し、線形計画問題として記述できるシミュレーション型多期間確率計画モデルを対象に議論する。

まずはじめに、金融工学における数学モデルを用いた問題解決プロセスについて議論する。そして、多期間最適資産配分問題に対するモデリング・プロセス(モデル化の方法)を述べる。シミュレーション型モデルでは、「投資配分比率を決めなければいけない問題」ではなく、「配分を決定する問題」であると捉え直すことによって、実際の取引を的確に記述でき、かつ、大域的最適解の導出を保証できるようにモデルを修正している。このようなモデリング技術を使って、モデルの定式化を行っている。一方、このタイプのモデルは不確実性の記述を詳細にするためには問題の規模は膨大になるが、問題の係数行列は疎大行列となる。一般に、疎大行列を持つ大規模線形計画問題は内点法を用いることによって高速解法が可能であり、汎用数理計画ソフトウェアを用いることによって、計算機上の実装もある程度容易である。しかし、シミュレーション型モデルの場合、制約式の係数行列の構造(非零パターン)が数本の密な(非零要素の多い)列を含むために、ナイーブな実装ではこの問題をうまく解くことができない。そこで、一般的な回避策として知られている工夫を含む様々な内点法の実装上の技法、特に一次方程式の解法に対する技法を工夫することによって計算時間を向上させることができる方法を示す。

1　はじめに

複数の投資対象の中から投資家にとって最も好ましいように、どの投資対象にどれだけ投資をしたらよいかという問題をポートフォリオ最適化問題という。ポートフォリオ最適化問題を解くためのモデルとしては、モデル構築や解法上の容易さから、運用期間にかかわらず、平均・分散モデルを代表とする1期間モデルが中心的に用いられている。しかし、年金基金などの長期的な資金運用を行う投資家にとって、多期間にわたる不確実性を考慮した動的投資政策の決定を「明示的に」モデル化するためには、1期間モデルではなく、多期間モデルを構築する必要がある。

多期間ポートフォリオ最適化問題を実際に解くためのモデルとしては、シナリオ・ツリーを用いた多期間確率計画モデルが中心となって発展している[1]。また、多期間ポートフォリオ最適化問題の基本的な枠組みとして最初に提示された確率制御モデルのタイプ[2]でも、モンテカルロ・シミュレーションにより発生させたパスを利用して不確実性を記述することによって、問題を解くためのモデル化が考えられる。このようなタイプのモデルはシナリオ・ツリーに比べて不確実性をより詳細に記述することが可能であるが、非凸非線形計画問題として定式化されるため、一般に大域的最適解の導出を保証することができない。そこで、枇々木[7]は、モンテカルロ・シミュレーションによるパスを用いて不確実性を記述した確率制御(動的確率計画)モデルの枠組みのもとで、従来想定している投資決定ルールを変更することによって線形計画問題[3]として記述できるモデル化を提案している。このタイプのモデルは「シミュレーション型多期間確率計画モデル」と呼ばれる。通常、ポートフォリオ最適化問題は投資比率を求める問題として記述されるが、多期間モデルではそのことがモデルの非線形構造の原因となっている。投資ルールを変更し、投資比率の代わりに投資額または投資量を求める問題に変更することによって、線形計画モデルとしての定式化を可能にしている[4]。

線形計画問題として定式化が可能になれば、大規模な問題でも大域的最適解の導出が保証され、実務的にも有効なモデル化になる[5]。シミュレーション型確率計画モデルは、その決定変数を投資比率と置いたときには、従来の確率制御(動的確率計画)モデルと同じタイプのモデルである。「投資配分比率を決めなければいけない問題」ではなく、「配分を決定する問題」であると捉え直すことによって、実際の取引を的確に記述でき、かつ、大域的最適解の導出を保証できるようにモデルを修正している。これは金融工学における重要な技術であるモデリング技術であると考えてよい。モデルは線形計画問題として定式化が可能であるため、大規模な問題に対する計算機上の実装もある程度容易である。ここで、「ある程度」と書いたのは、ナイーブな実装ではこの問題を高速に解くことができないからである。この問題に対する適切な解法の選択を行うためには問題の

[1] シナリオ・ツリーを用いた多期間確率計画モデルは近年、コンピュータの高速化と解法アルゴリズムの発展に伴い、大規模な問題を解くことが可能になり、様々な研究が行われている。詳細は、Mulvey and Ziemba[20, 21]の参考文献を参照されたい。しかし、シナリオ・ツリー型モデルは不確実性の記述を詳細にしようとすると、問題の規模が指数的に増加するという欠点がある。また、問題を大規模にしないためには数少ないシナリオでうまく不確実性を記述しなければいけない難しさもある。

[2] Merton[16] が連続時間モデル、Samuelson[22] が離散時間モデルを示した。

[3] 目的関数が非線形凸関数の場合には、非線形凸計画問題となるが、大域的最適解の導出は保証される。

[4] シナリオ・ツリー型モデルでも投資比率を用いると非線形計画問題となるので、投資額もしくは投資量を求める問題として定式化される。シナリオ・ツリー型モデルでは、投資比率、投資額、投資量のいずれを用いてもそれらの定式化は等価になるのに対し、シミュレーション型では3種類の定式化は等価にはならない。

[5] 実務的には、常に安定して最適解が導出されることは極めて重要な要素である。非凸非線形計画問題では、ある特別なクラスの問題を除いて、一般に大域的最適解の導出は保証されないので、初期点によって求められる解が異なる(局所的最適解になる)可能性があり、初期点を様々に変更して問題を解かなければならない。

性質に関する理解と、アルゴリズムや実装に関する知識がある程度必要となる。そこで、本研究では実務的に利用できるモデルを構築し、問題を解決するためのプロセスについて述べるとともに、モデリング技術、解法の選択、アルゴリズム、実装技術の重要性について議論する。

ところで、1期間モデルは株式ポートフォリオ運用や資産配分決定など、様々なタイプの問題に適用されている。多期間モデルも定式化上はほとんどの問題に適用することは可能であるが、実際には問題の規模が膨大になるために、現時点では資産配分問題やALM問題などへの適用が現実的である。以降では、資産配分問題への適用を中心に多期間確率計画モデルについて議論していく。

本論文の構成は以下の通りである。2節では、金融工学においてモデリング技術と実装(計算)技術の重要性を確認するために、金融工学における数学モデルを用いた問題解決プロセスについて多期間資産配分モデルの例も挙げて説明する。3節では、多期間最適資産配分問題に対するモデリング技術の重要性について述べる。4節では、解法の選択と内点法アルゴリズムにおける実装技術について説明する。最後に、5節で結論と今後の課題を述べる。

2　金融工学における問題解決プロセス

問題を数学モデルによって解決するためのプロセスは図1のように記述できる。

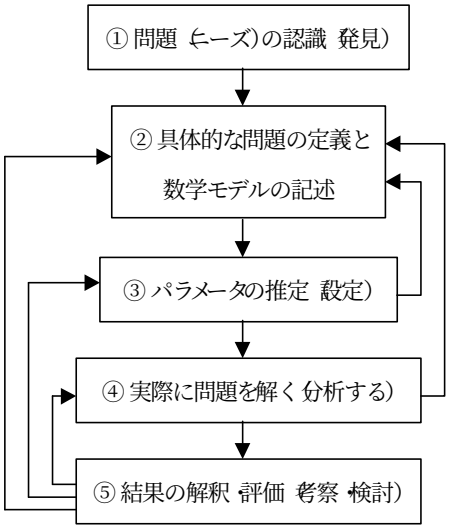

図 1：数学モデルを用いた問題解決プロセス

各フェーズの意味と必要な知識や技術について説明する。

第1フェーズ: 問題(ニーズ)を認識（発見）する。

金融工学で取り扱う必要のある問題には、投資信託や年金基金などの資産運用、銀行や保険会社、年金基金などのALM、市場リスクや信用リスクの管理、派生証券の価格付けなど、金融市場や金融取引における様々な問題がある。これらの問題(ニーズ)は金融業界において十分に認識されているが解決されていないものから、今のところ認識されていないが今後問題になることが予想される問題まで様々である。これらの問題を認識し、発見するためには、金融理論に関する基本的な知識や経験が必要であるとともに、問題をとらえるためのフレームワークを常に意識して

いなければならない。

第2フェーズ: 具体的な問題を定義し、数学モデルとして記述する。

認識(発見)された問題はある程度具体化された問題から漠然と捉えられている問題まで様々である。まず漠然とした問題に関しては具体化する作業が必要である。そして、具体化された問題を数学モデルとして記述する。数学モデルの記述レベル(記述の厳密さ)は問題や最終的に得たいと考えている解の正確さに依存する。問題の本質をゆがめない範囲で、しかも第3フェーズのパラメータ推定や第4フェーズにおける解の導出のことを考慮した形でいかにうまい数学モデルを記述できるかがキーポイントになる。はじめからうまい数学モデルが作れるとは限らないので、第2フェーズから第4フェーズはフィードバックを繰り返す必要がある。このフェーズを機能させるためには、問題設定能力に加えて、金融の知識やモデリング技術など幅広い知識が要求される。

モデリング技術とは、例えば、数理計画モデルを構築する場合には、その解法である数理計画法の基礎知識はもちろんのこと、どのようにモデルを作成すれば、

- 問題の本質をきちんと捉えられるか、
- 実際に問題を解くことができるか(アルゴリズムは開発できるか、もしくは開発されているか)、

など、モデル化に関する十分な知識、知恵、経験をもとにモデル構築を行うことができる技術である。モデリング技術は、一般的に定義することは難しく、そのため認識・評価されにくいが、今後は実際に問題解決を行うために必要な技術として明示的に認識する必要がある。

第3フェーズ: モデルに用いるパラメータを推定(設定)する。

数学モデルに用いるパラメータが不確実な場合には、統計手法を用いて確率分布を記述するパラメータを推定する。そのためには、確率過程、計量経済学、統計解析などの手法を理解するとともに、パラメータ推定に関するモデリング技術が必要となる[6]。最近は市販の統計パッケージが使いやすくなっており、一般的な統計手法であれば、手軽に利用でき、特に統計理論に関する専門家でなくても実際にパラメータ推定を行うことができる。

第4フェーズ: 実際に問題を解く(分析する)。

推定(設定)されたパラメータを用いて、数学モデルによって実際に問題を解く場合、一般的によく知られた方法や市販のソフトウェアを用いることによって問題を解くことができる。しかし、多くの計算時間がかかる問題もあり、その場合にはその問題に固有の構造を利用したアルゴリズムを用いることは非常に効果的である。例えば、複雑なペイオフを持つ派生証券の価格付けにはモンテカルロ・シミュレーションが用いられるが、分散削減法やLDS(超一様分布列)を使うことが計算効率に多大な影響を与える問題もある。また、大規模な数理計画問題の場合には疎大行列になることが多く、行列構造によっては、それをうまく利用して高速に問題を解くアルゴリズムが数理計画法の分野で開発されているし、非凸計画問題に対する様々な大域的最適化の方法論も開発されている。このように、アルゴリズムの開発技術と利用技術、計算機への実装技術(ソフトウェア開発・利用技術)は、実際の解を求める上で、極めて重要である。

その一方で、数理計画問題に対しては、数理計画法のソフトウェアを用いることによって高速

[6]パラメータ推定問題は、より具体化された金融工学における一つの問題である。

に問題を解くことができ、ソフトウェアにはモデル化した数式を手軽に記述できるモデル記述言語[7] が付いている。特別なアルゴリズムを実装しているソフトウェアも増えてきており、特殊な問題でなければ、特に第 4 フェーズに必要な技術を意識することなしに問題を解くこともできる環境が整い始めている。

第 5 フェーズ: 結果を解釈し、評価 (考察・検討) する。

得られた最適解から具体的な方策を導いたり、数学モデルが問題解決にどのような役割を果たすことができたかなどを評価し、様々な知見を得るためには、金融工学周辺の総合的な知識と関連技術を理解する能力が必要である。

上記の 5 つのフェーズを機能させるためには、他のフェーズ (特に前後のフェーズ) をある程度理解でき、かつそのフェーズに精通した人 (スペシャリスト) と全体的な各フェーズのつながりを理解できる人 (コーディネータ) の存在が不可欠である[8] 。すべてのフェーズを 1 人で行うことができれば、各フェーズを意識する必要はないし、ある程度モデルやソフトウェアなどに関する既存の知識によって代替できる場合にも 1 人で問題解決することは難しくない。しかし、広範な知識や、より高度で複雑な技術を要求される場合には 1 人で行うことは難しいだろう[9] 。本論文で議論する多期間最適資産配分問題はその一例である。この問題解決のフェーズを以下に示す。

① 年金基金運用、投信会社、保険会社の資金運用など、長期的な資産運用を多期間にわたる様々な点を考慮して科学的手法に基づき行う問題を対象とする。

② 多期間にわたる最適投資量を求める問題として定義し、確率計画モデルとして、記述する。第 3 フェーズ、第 4 フェーズを考慮した問題設定と数理計画モデリングおよびそのための金融知識が必要である。

③ 将来の収益率やキャッシュ・フローなどを推定 (設定) する。パラメータ推定を行うための技術が必要である。

④ 実際に多期間確率計画問題を解く (分析する)。第 2 フェーズのモデルに合わせた実装技術やソフトウェア利用技術が必要である。

⑤ 結果を解釈し、評価 (考察・検討) する。多期間資産運用に関する総合知識と関連技術の理解が必要である。

本論文では、すでに問題を認識していて (第 1 フェーズ)、パラメータ推定 (第 3 フェーズ) [10] に関しては既存のモデルを利用することにし、第 3節で第 2 フェーズ、第 4節で第 4 フェーズを順次議論する。

[7] モデル記述言語は数学モデルを直感的に、そして、ほぼ直接的に記述できる。また、モデリング記述言語の利点はモデルの修正が簡単なのに加えて、モデルを一度作り固定してしまえば、データを変更するだけで定型的な処理をすることもできることである。つまり、モデリング記述言語はモデル構築に関する限りは通常のプログラミング言語で記述することと同じようなプログラムを生成できる。

[8] 各フェーズのスペシャリストが他のフェーズを理解する能力があまりないときほど、コーディネータの存在は重要である。

[9] 第 5 フェーズは第 1 フェーズから第 4 フェーズのスペシャリストとコーディネータが集まって行うことになるだろう。

[10] 第 3 フェーズは、ファイナンス理論において蓄積された、もしくは将来出てくるであろう研究成果を生かすことができる。第 2 フェーズではこれらの研究成果を利用しやすいモデル化を行う。

3　モデリング技術の重要性

本論文で議論するモデリング技術を、「定性的に記述される問題を定量的に、しかも実際に効率よく解を求めることができるような数学モデルとして記述できる技術」と定義する[11] 。ここでは、以下の3点からモデリング技術について議論する。

- 投資比率を用いた多期間資産配分問題のモデル化
- 投資量を決定変数とするモデル化の必要性
- シミュレーション型モデルの構築

3.1　投資比率を用いた多期間資産配分問題のモデル化

n 個の危険資産 $(j = 1, \ldots, n)$ と現金 $(j = 0)$ に資金を配分する問題を考える。資産0を現金(安全資産、コールローン)、資産1 〜 資産n を危険資産とする対象資産数が $n + 1$ 個の資産配分問題である。0時点を投資開始時点、T 時点を計画最終時点とする。また、$t - 1$ 時点から t 時点までの期間を期間 t とする。

多期間計画モデルにおいて、期間 t の収益は $t - 1$ 時点の投資配分によって決定される。モデル化を行う際に注意しなければいけないことは、危険資産は t 時点にならないと期間 t での収益が確定しないのに対し、現金(安全資産)の期間 t における収益は $t - 1$ 時点(投資配分を行う時点)で確定することを考慮することである。

以降のモデル化は多期間モデルに不可欠な定式化のみを示す[12] 。一般に、ポートフォリオ選択問題における投資の意思決定は「投資比率」を決定することであるので、ここではその定式化を示す。まず、はじめに、モデル化に用いる記号を示す[13] 。

(A) パラメータ

$\tilde{\mu}_{jt}$: 期間 t の危険資産 j の投資収益率。$(j = 1, \ldots, n;\ t = 1, \ldots, T)$

r_0 : 期間 1 の金利(0 時点のコール・レート)。

$\bar{r}_{t-1}$: 期間 t の金利($t - 1$ 時点のコール・レート)。$(t = 2, \ldots, T)$

W_0 : 0時点での富(初期富)。

(B) 決定変数[14]

w_{jt} : t 時点の危険資産 j への投資比率。$(j = 1, \ldots, n;\ t = 0, \ldots, T - 1)$

c_t : t 時点の現金(コール運用)の比率。$(t = 0, \ldots, T - 1)$

ここで、'〜'は確率変数を表す。金利も確率変数であるが、各投資決定時点で確定するので、'—'を付けている[15] 。

[11] 広義の意味では、数学モデルに限る必要はないが、本論文では、具体的に問題解決をするための技術としてとらえるために、ある程度狭義の意味で示す。
[12] 実際には、投資比率の上下限制約や売買回転率制約などがこれらの定式化に追加される。
[13] モデルに対する入力変数をパラメータ、モデルからの出力変数を決定変数とする。
[14] t 時点での富 $\tilde{W}_t$ $(t = 1, \ldots, T)$ は w_{jt} と c_t によって記述される。
[15] 期待値ではないことに注意すること。

ポートフォリオの収益(富)は、「ポートフォリオの構成」と「ポートフォリオに含まれる資産価格変動(収益率)の確率分布」によって決まる。各資産価格変動の確率分布はあらかじめ想定された分布を用いるが、各時点でのポートフォリオの収益分布(富の分布)はポートフォリオの構成によって様々に変化する。多期間にわたる最適資産配分問題は、投資家にとって最大の期待効用、もしくは好ましいリスク・リターンを得られるように、各時点でのポートフォリオの収益の確率分布を制御する動的な資産配分を決定する問題である。以下に、配分決定のための制約条件式とそれを用いて計算する各時点の富の計算式を示す。

(1) 0時点での配分決定：$w_{j0},\, c_0$

$$\sum_{j=1}^{n} w_{j0} + c_0 = 1 \tag{1}$$

(2) 0時点の配分決定による1時点の富：$\tilde{W}_1$

0時点で決定されたポートフォリオの(1時点までの)1期間における収益率 $\tilde{\mu}_{p1}$ は、

$$\tilde{\mu}_{p1} = \frac{\tilde{W}_1}{W_0} - 1 = \sum_{j=1}^{n} \tilde{\mu}_{j1} w_{j0} + r_0 c_0 \tag{2}$$

となり、(2)式を変形すると、(3)式を得る。

$$\tilde{W}_1 \;=\; (1+\tilde{\mu}_{p1})\, W_0 = \left\{ \sum_{j=1}^{n} (1+\tilde{\mu}_{j1}) w_{j0} + (1+r_0) c_0 \right\} W_0 \tag{3}$$

(3) $t-1$ **時点の配分決定(**$w_{j,t-1}$**,** c_{t-1}**)による** t **時点の富**：$\tilde{W}_t \;(t = 2, \ldots, T)$

同様に、t 時点の富は $t-1$ 時点での配分決定をもとにして(4)式によって表すことができる。

$$\tilde{W}_t \;=\; \left\{ \sum_{j=1}^{n} (1+\tilde{\mu}_{jt}) w_{j,t-1} + (1+\bar{r}_{t-1}) c_{t-1} \right\} \tilde{W}_{t-1} \tag{4}$$

$$(t-1\text{時点の配分決定}) \quad \sum_{j=1}^{n} w_{j,t-1} + c_{t-1} = 1 \tag{5}$$

特に、$t = T$ のときは、計画最終時点(T 時点)における富を表し、(6)式で表すこともできる。

$$\tilde{W}_T \;=\; W_0 \cdot \prod_{u=1}^{T} \left\{ \sum_{j=1}^{n} (1+\tilde{\mu}_{ju}) w_{j,u-1} + (1+\bar{r}_{u-1}) c_{u-1} \right\} \tag{6}$$

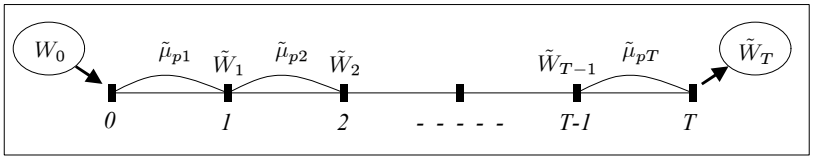

図 2：多期間投資決定問題のモデル化

(4) 目的関数

2つのタイプの目的関数を示す。

① 期待効用最大化

$$\textbf{Maximize} \quad E\left[U(\tilde{W}_T)\right]$$

② 2パラメータ・アプローチ

$$\textbf{Minimize} \qquad g[\tilde{W}_T]$$

$$\textbf{subject to} \qquad E[\tilde{W}_T] \geq W_E$$

ここで、$g[\tilde{W}_T]$ はリスク関数、W_E は投資家の要求する期待富を表す。

3.2 投資量を決定変数とするモデル化の必要性

制約式の中には確率変数を含む線形関数の乗法形式が含まれるため、実際にこのような問題を直接的に解くのは難しい。確率変数は離散化することによって取り扱いやすくなるが、それでも線形関数の乗法形式が離散化した数だけ制約式に含まれる。したがって、実際に問題を解くためには近似モデルを考える必要がある。近似モデルとして発展しているシナリオ・ツリー型モデルは、変数変換によって投資比率を決定変数とする定式化を投資額もしくは投資量を決定変数とする定式化に直し、非線形制約式を取り除いている。シナリオ・ツリー型モデルでは、投資比率、投資額、投資量のいずれを用いてもそれらの定式化は等価となるため、結果的には投資比率を決定変数とする問題を解くことができる[16] 。これはモデリング技術によって問題のクラスを簡単にした典型的なケースである。しかも、コンピュータの高速化と解法アルゴリズムの発展に伴い、大規模な問題を解くことが可能になった。このシナリオ・ツリー型モデルの定式化の方法には、分割変数表現とコンパクト表現の2種類の方法がある。コンパクト表現は分割変数表現に比べて、異時点間の関係を明示的に記述するモデル化のために多少取り扱いにくいが、制約式の本数と決定変数の数 (問題の規模) ははるかに小さい。単純に考えると、コンパクト表現の方が問題の規模が小さいので良いように感じるが、分割変数表現による定式化は疎大構造を持っているので、内点法の計算アルゴリズムにとても適しているという研究結果もある。一方、コンパクト表現をすることによって内点法では有利にはならないが、単体法では有利であるという研究もある。確率計画モデルでは、モデルとアルゴリズムが密接に関連しており、問題の定式化が異なれば、異なるアルゴリズムが必要となる。このような等価な2種類の定式化の優劣を比較するためには、アルゴリズムを理解していないとできない一種のモデリング技術と言えるだろう。詳しくは、Messina and Mitra[17] を見よ。

一方、詳細なモデルについては次節に譲るが、シミュレーション型モデルでも、決定変数を投資比率としている限り、非凸非線形計画問題として定式化され、一般に大域的最適解の導出を保証することができない。また、シナリオ・ツリー型モデルとは異なり、変数変換を行うことによって投資額もしくは投資量を決定変数とした等価なモデルを記述することができない。そこで、「投資配分比率を決めなければいけない問題」ではなく、「配分を決定する問題」として捉え直し、実際の取引を的確に記述でき、かつ、大域的最適解の導出を保証できるように、投資額もしくは投資量を決定変数として、改めてモデル化を行った[17] 。具体的には、モデルの基本構造を表す制約

[16] 詳細な書き換えについては、枇々木 [6] 第7章を参照されたい。

[17] 投資比率、投資額、投資量のどれを決定変数にするかによってモデルから得られる最適解は異なる (等価にならな

式はすべて線形式となるので、凸計画問題(目的関数が線形ならば、線形計画問題)として記述が可能となる。

3.3 シミュレーション型モデルの構築

3.3.1 投資の意思決定とモデル化

シナリオ・ツリー型モデルでは不確実性の記述を詳細にしようとすると、問題の規模が指数的に増加するという欠点があり、その欠点を克服するためのモデルがシミュレーション型モデルである。対象とする資産数が増加する(次元数が多くなる)と、シナリオ・ツリーにおいては、時間の経過とともに分岐数も急速に増加するため、数少ない分岐数で、しかもうまく不確実性を記述しなければならない。資産間や異時点間の関係を考慮すると、そのような収益率モデルの記述は難しくなる。それに対し、シミュレーション型モデルでは、図3のようなシミュレーション経路によって、将来の資産価格の不確実な変動を記述する。資産価格変動(収益率)の振る舞いが確率微分方程式(確率差分方程式)や時系列モデル式などで記述されるとき、モンテカルロ・シミュレーションによって複数のサンプル・パス(シミュレーション経路)[18]を生成するのは比較的容易であり、シナリオ・ツリーに比べて不確実性をより詳細に記述することが可能である。

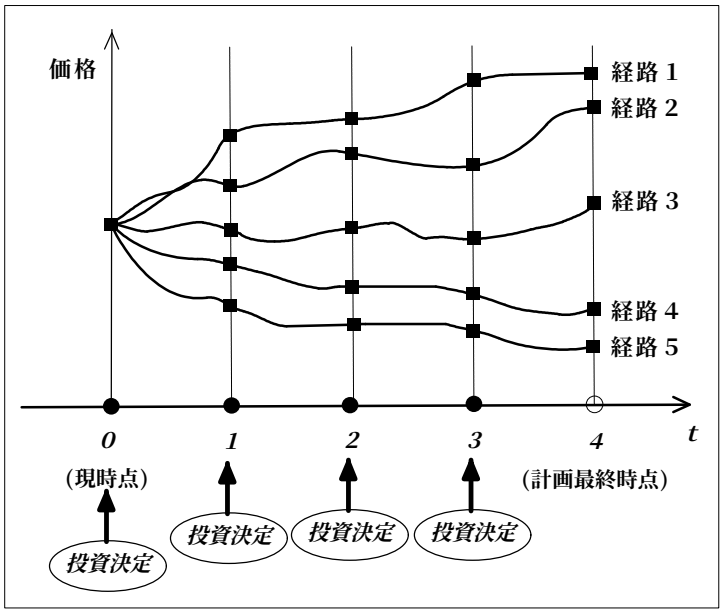

図 3 : シミュレーション経路と投資の意思決定

どのような確率分布に従っている場合でも(確率分布を複雑に合成した場合でも)、シミュレーションで生成された複数のサンプル・パス(シミュレーション経路)を直接的に利用することができるので、極めて柔軟で強力なモデル化の方法であるが、投資の意思決定を記述する上では、工夫が必要となる。なぜならば、シミュレーション経路は1本の経路にだけ注目すると、t 時点の状態の次に発生する $t+1$ 時点の状態は1つしか想定しないため、シナリオ・ツリー型モデルのように、各状態ごとに投資の意思決定を別々に行うと、それは確定条件下での意思決定を行うことに

い)ことに注意が必要である。詳細な定式化は、枇々木[6]第8章を参照されたい。

[18]シナリオ・ツリー上の経路(シナリオ・パス)との違いやモンテカルロ・シミュレーションによる経路生成を強調するために、シミュレーション経路と呼ぶ。

なるからである。そこで、シミュレーション型モデルの場合には、すべての時点で状態に依存しない取引戦略による意思決定を行わなければならない[19] 。既知のシミュレーション経路上で、逐次的に各時点において前時点の意思決定の条件のもとで意思決定を行うことになる[20] 。

3.3.2 モデルの定式化

投資量を決定変数とし、2パラメータ・アプローチによる下方リスクモデルの定式化を示す。リターン尺度には計画最終時点における富(最終富 $\tilde{W}_T$)の期待値(期待富)、リスク尺度には計画最終時点における富がその目標水準(目標富 W_G)を下回る大きさ(不足分)を示す1次の下方部分積率[21] を設定する。はじめに用いる記号を示す。

(A) パラメータ

I ：経路の本数

ρ_{j0} ：0時点の危険資産 j の価格。$(j = 1, \ldots, n)$

$\rho_{jt}^{(i)}$ ：t 時点の経路 i の危険資産 j の価格。$(j = 1, \ldots, n;\ t = 1, \ldots, T;\ i = 1, \ldots, I)$

r_0 ：期間 1 の金利(0 時点のコール・レート)。

$r_{t-1}^{(i)}$：期間 t の経路 i の金利($t-1$ 時点のコール・レート)。$(t = 2, \ldots, T;\ i = 1, \ldots, I)$

W_0：0時点での富(初期富)。

W_E：計画最終時点で投資家が要求する期待富。

W_G：計画最終時点での目標富。

(B) 決定変数

z_{jt} ：t 時点の危険資産 j への投資量。$(j = 1, \ldots, n;\ t = 0, \ldots, T-1)$

v_0 ：0 時点の現金(コール運用額)。

$v_t^{(i)}$：t 時点の経路 i の現金(コール運用額)。$(t = 1, \ldots, T-1;\ i = 1, \ldots, I)$

$q^{(i)}$：計画最終時点の経路 i の富の目標富に対する不足分。$(i = 1, \ldots, I)$

---

[19]状態に依存しないとは、どの状態に到達するかに関わらず、一つの投資決定をすべてのシミュレーション経路に適用することを意味する。ただし、現金は運用する各時点で収益が確定するので状態に依存してもよい。投資の意思決定が経路にかかわらずに各時点で1通りしか許されないのは、非予想条件(non-anticipativity condition)を保つためである。非予想条件とは、モデルの定式化において、将来の不確実な状態の中からどの状態が生じるかを確定的に知っていることを利用して意思決定ができる機会を許さない条件のことである。不確実性下の投資決定を行う確率計画モデルには必要な条件である。

[20]枇々木[8]は、シミュレーション経路を用いつつ、条件付き意思決定を可能にしたモデルを提案している。このモデルは、シミュレーション/ツリー混合型モデルと呼ばれ、シナリオ・ツリー型モデルとシミュレーション型モデルの2種類のタイプのモデルの長所をうまく組み合わせた形で定式化されている。詳しくは、[8]を参照されたい。

[21]1次の下方部分積率は

$$LPM_1 = E\left[\left|\tilde{W}_T - W_G\right|_-\right] = \frac{1}{I}\sum_{i=1}^{I}\left|W_T^{(i)} - W_G\right|_-$$

と表すことができる。ここで、$|a|_- = \max(-a, 0)$ である。また、$W_T^{(i)}$ は 計画最終時点の経路 i の富を示す。定式化においては、(7), (12), (16)式を用いて計算される。

(C) 定式化

モデルは以下のように定式化できる。

$$\textbf{Minimize} \quad \frac{1}{I}\sum_{i=1}^{I} q^{(i)} \tag{7}$$

subject to

$$\sum_{j=1}^{n} \rho_{j0} z_{j0} + v_0 = W_0 \tag{8}$$

$$\sum_{j=1}^{n} \rho_{j1}^{(i)} z_{j1} + v_1^{(i)} = \sum_{j=1}^{n} \rho_{j1}^{(i)} z_{j0} + (1+r_0)v_0 \quad \left(= W_1^{(i)}\right), \quad (i = 1, \ldots, I) \tag{9}$$

$$\sum_{j=1}^{n} \rho_{jt}^{(i)} z_{jt} + v_t^{(i)} = \sum_{j=1}^{n} \rho_{jt}^{(i)} z_{j,t-1} + \left(1 + r_{t-1}^{(i)}\right) v_{t-1}^{(i)} \quad \left(= W_t^{(i)}\right),$$
$$(t = 2, \ldots, T-1;\ i = 1, \ldots, I) \tag{10}$$

$$\sum_{j=1}^{n} \overline{\rho}_{jT} z_{j,T-1} + \frac{1}{I}\sum_{i=1}^{I} \left(1 + r_{T-1}^{(i)}\right) v_{T-1}^{(i)} \geq W_E \tag{11}$$

$$\left\{ \sum_{j=1}^{n} \rho_{jT}^{(i)} z_{j,T-1} + \left(1 + r_{T-1}^{(i)}\right) v_{T-1}^{(i)} \right\} + q^{(i)} \geq W_G, (i = 1, \ldots, I) \tag{12}$$

$$z_{jt} \geq 0, \quad (j = 1, \ldots, n;\ t = 0, \ldots, T-1) \tag{13}$$

$$v_0 \geq 0 \tag{14}$$

$$v_t^{(i)} \geq 0, \quad (t = 1, \ldots, T-1;\ i = 1, \ldots, I) \tag{15}$$

$$q^{(i)} \geq 0, \quad (i = 1, \ldots, I) \tag{16}$$

ここで、$\overline{\rho}_{jT} z_{j,T-1}$ は危険資産 j の T 時点の価格の期待値である。また、(9), (10) 式の値 $W_t^{(i)}$ は t 時点の経路 i の富 $(t = 1, \ldots, T;\ i = 1, \ldots, I)$ を表す。投資量を決定変数にしたモデルを解くことによって、危険資産 j への投資額および投資比率も以下のように計算することができる[22] 。

【 投資額 】

- **0 時点の危険資産 j への投資額** : $\rho_{j0} z_{j0}$
- **t 時点の経路 i の危険資産 j への投資額** : $\rho_{jt}^{(i)} z_{jt}$

【 投資比率 】

- **0 時点の危険資産 j への投資比率** : $\frac{\rho_{j0} z_{j0}^*}{W_0}$
- **0 時点の現金 (コール運用) の比率** : $\frac{v_0^*}{W_0}$
- **t 時点の経路 i の危険資産 j への投資比率** : $\frac{\rho_{jt}^{(i)} z_{jt}^*}{W_t^{(i)*}}$
- **t 時点の経路 i の現金 (コール運用) の比率** : $\frac{v_t^{(i)*}}{W_t^{(i)*}}$

[22] **投資額および投資比率は経路 i によって様々な値になるので注意すること。**

(D) モデル記述言語 SIMPLE による記述例

数理計画問題を解くためにソフトウェアを用いることは不可欠である。市販の汎用ソフトウェアでは、数理計画モデルを数式のイメージどおりに記述するための言語であるモデル記述言語を利用することができる[23] 。数理計画法パッケージ NUOPT [24] **のモデル記述言語である** SIMPLE **を用いて、モデルの定式化部分を記述すると図**4**のようになる。**

```
// 集合の定義
  Set Path; Element i(set=Path);
  Set Asset; Element j(set=Asset);
// パラメータの定義
  Parameter T(name="T"); // 計画期間
  Parameter I(name="I"); // 経路の本数
  Parameter rho0(name="rho0",index = j); // 0時点の危険資産 j の価格
  Parameter rho(name="rho",index = (j,et,i)); // et 時点の危険資産 j の価格
  Parameter r0(name="r0"); // 期間1の金利(0時点のコールレート)
  Parameter r(name="r",index = (dt1,i)); // 期間dt+1の金利(dt時点のコールレート)
  Parameter W0(name="W0"); // 初期富
  Parameter we(name="we"); // 要求期待富
  Parameter wg(name="wg"); // 目標富
// 変数の定義
  Variable z(name="z",index = (j,dt0)); // dt0 時点の危険資産 j の投資量
  Variable v0(name="v0"); // 0時点・現金(コール運用)
  Variable v(name="v",index = (dt1,i)); // dt1 時点・現金
  Variable q(name="q", index = i); // 最終富・不足分
// 制約式
  sum(rho0[j]*z[j,0],j) + v0 == W0; // 0時点・資産配分
  sum(rho[j,1,i]*z[j,1],j) + v[1,i] == sum(rho[j,1,i]*z[j,0],j) + (1+r0)*v0; // 1時点C.F.等式
  sum(rho[j,dt2,i]*z[j,dt2],j) + v[dt2,i] ==  sum(rho[j,dt2,i]*z[j,dt2-1],j)
      + (1+r[dt2-1,i])*v[dt2-1,i];                                    // dt2時点C.F.等式
// 非負条件
  z[j,dt0] >= 0;
  v0 >= 0;
  v[dt1,i] >= 0;
  q[i] >= 0;
// 最終富の期待値の計算と制約式
  Expression MeanRho(index=(j,et)); MeanRho[j,et]=sum(rho[j,et,i],i) /I;
  sum(MeanRho[j,T]*z[j,T-1],j) + sum((1+r[T-1,i])*v[T-1,i],i)/I >= we;
// リスクに関する制約式
  sum(rho[j,T,i]*z[j,T-1],j) + (1+r[T-1,i])*v[T-1,i] + q[i] >= wg;
// 目的関数
  Objective Risk(name="Risk",type=minimize); Risk=sum(q[i],i)/I;
  solve();
```

図 4 : **シミュレーション型モデルの** SIMPLE **による記述例**

[23]**モデリング記述言語の利点はモデルの修正が簡単なのに加えて、モデルを一度作り固定してしまえば、データを変更するだけで定型的な処理をすることもできることである。つまり、モデリング記述言語はモデル構築に関する限りは通常のプログラミング言語で記述することと同じようなプログラムを生成できる。**

[24]NUOPT **は(株)数理システムの製品である。**4**節の最適化計算の結果はすべて** NUOPT **を用いた場合である。**

4　解法の選択・アルゴリズム・実装技術

3節において、定式化およびそのモデリング言語による問題の記述を行った。本節では実際に問題を解き、数値的な解を求める第4フェーズについて説明する。

問題の記述と求解は概念上独立した操作である。ここでは記述された問題を一般化して、汎用アルゴリズムを用いるソフトウェアによって求解するアプローチを採る。巡回セールスマン問題など、一部の組み合わせ問題のように非常に求解が困難な場合には、問題記述と解法を密接に連関させ、専用アルゴリズムやそれを用いたソフトウェアを作成する場合もある。しかし、シミュレーション型多期間確率計画モデルは線形計画問題として記述でき、強力な汎用アルゴリズムを装備したソフトウェアが現に利用可能であることから、あえて問題の記述と求解を独立させるアプローチを採用した[25] 。このアプローチの利点としては以下が挙げられる。

(1) 問題の記述に関する柔軟性を獲得できる。

(2) 汎用アルゴリズムを装備したソフトウェアの進歩を織り込むことができる。

4.1　アルゴリズム

一般的な線形計画問題

$$\begin{aligned} &\text{最小化} \ : \ \boldsymbol{c}^t\boldsymbol{x} \\ &\text{制約} \quad : \ A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \mathbf{o} \end{aligned} \tag{17}$$

に対する解法として現在普及しているものには、単体法と内点法の二つが知られている。単体法は変数を非基底変数、基底変数として区分けして、この区分けを変更しながら最適解に近づく。対して内点法は変数の区分けを連続変数の大小関係によって表現してニュートン法と類似の方法で解を更新しながら最適解を求める方法である。

内点法は変数やアルゴリズム中に現れるパラメータの更新方法について異なる様々な求解アルゴリズムが発表されており、大域的収束性[26]や局所的収束性[27]などの見地から計算複雑度の解析が行われている。そのうちのいくつかは単体法よりも低い計算オーダーで十分な精度の解が得られることが理論的に保証されている。しかし、現存しているソフトウェアによる実際の計算量は理論的に与えられる上界をはるかに下回っているため、これらの理論的解析は具体的な問題に対する優劣を直接結論付けるものではない。

[25] もちろん、問題に対して適切な解法の選択を行うためには問題の性質に関する理解と、アルゴリズムや実装に関する知識がある程度必要であることは言うまでもない。

[26] 初期値によらずに解に収束する。

[27] 解の近傍で十分短い時間で解に収束する。

4.2 実装技術の影響

シミュレーション型多期間確率計画モデルから生成される線形計画問題の規模は表1のようになる。

表 1：多期間確率計画モデルの規模

変数総数	$(n+I)T+1$
制約式総数	$TI+2$

ここで、n は危険資産数、I は経路の本数、T は期間数である。モンテカルロ・シミュレーションによって経路を発生することを考えると、経路の本数 I は一般に大きく、実際の応用例ではこの問題は変数、制約式の数ともに数万の大規模問題になる。アルゴリズムに関する一般的考察から、解法には内点法を用いるのが適当と考えられる。比較的小規模な問題 ($T=3$, $n=3$, $I=500$:変数総数$=1,510$, 制約式総数$=1,502$) に対する表2の実験結果からもそれは裏付けられる。

表 2：$T=3$, $n=3$, $I=500$ の場合の計算時間

単体法	20 秒
内点法	5 秒

(使用計算機：Pentium300MHz, メモリ 256M バイト)

ところで、表2の内点法に対する実験結果は 4.3項で述べるように、密な列の扱いに実装上の工夫を行った場合の結果である。この実装上の工夫を無効にすると内点法と単体法の優劣は表3のように逆転する。

表 3：実装上の工夫を行わない場合の求解時間

単体法	20 秒
内点法 (DS 回避なし)	60 秒

(使用計算機：Pentium300MHz, メモリ 256M バイト)

この理由は、比較的少数個の変数が非常に多くの制約式に現れているためである。これらの変数に対応する係数行列の列には非零要素が数多く現れ、「密な列」と呼ばれる。3.3.2項 (D) の定式化における制約式の係数行列を図5に示す。

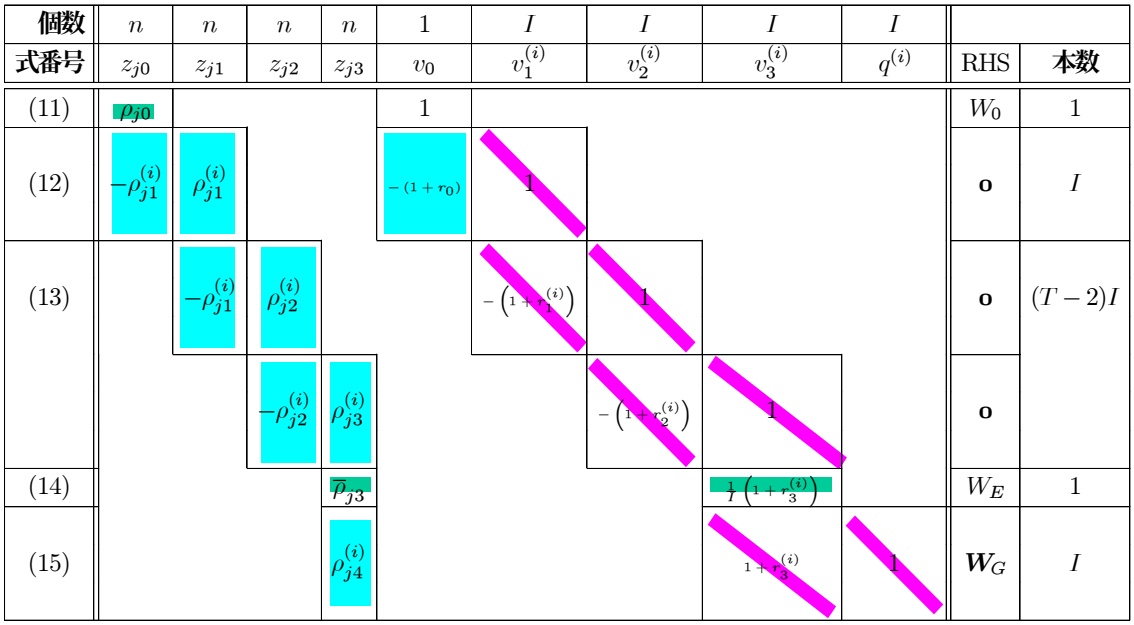

図 5：制約式の係数行列の構造

危険資産への投資量 (z_{jt}) の部分に密な列が含まれている。内点法の実装においては、解法内部で取り扱う係数行列が疎、すなわちほとんどの要素が 0 であることを最大限利用することが高速化の鍵である。この実験結果は密な列を含む問題に対して工夫を行わなければ係数行列の疎行列性が破壊され、計算量が急増してしまうことを示している。

このように解法の選択にはアルゴリズムの一般的な性質のみならず、その実装 (ソフトウェア) が拠っている技法が大きな意味を持つ。もとの実装で経路の数を増やした際の結果は表 4のようになり、内点法アルゴリズムがその特質を発揮していることがわかる。

表 4：経路数を増やした場合の実験 ($T = 3$, $n = 3$)

経路数	5,000	10,000
変数総数	15,010	30,010
制約式総数	15,002	30,002
単体法	1,940 **秒**	—
内点法	82 **秒**	340 **秒**

(**使用計算機**：Pentium400MHz, **メモリ** 128M **バイト**)

いくつかのケースで計算した結果を図 6に示す。ケース 1 はリスク最小化問題、ケース 2 は期待最終富最大化問題、ケース 3 とケース 4 はある期待富のもとでのリスク最小化問題を解いている。ケース 5 は資産への組み入れ比率や売買回転率に上限を設定した場合の問題を解いている。問題の設定方法によって計算時間は異なる。

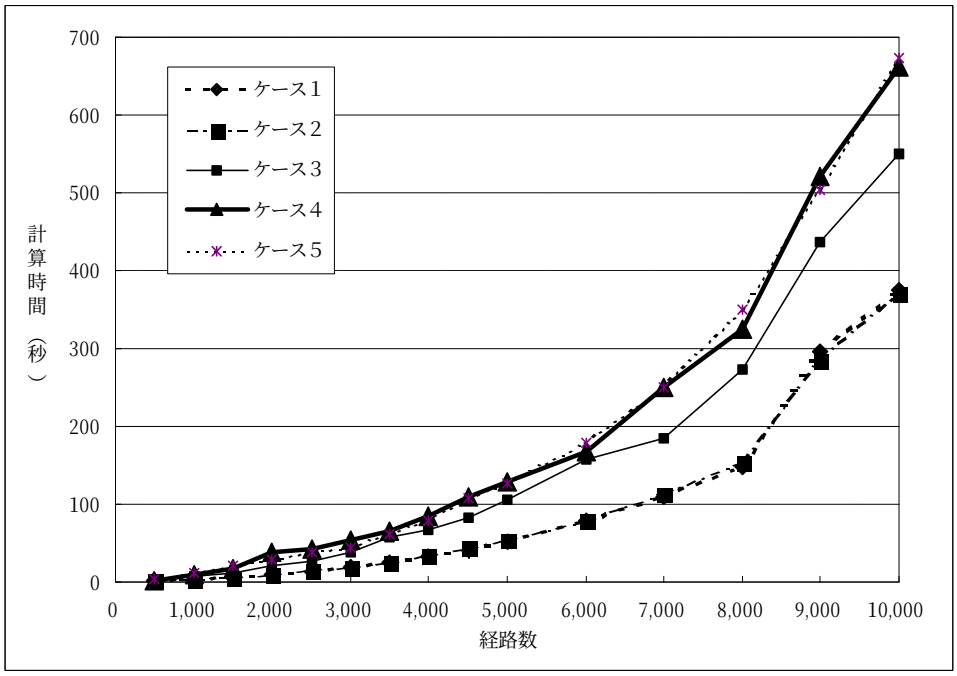

図 6：経路数の違いによる内点法の計算時間

4.3 内点法の実装技術

内点法に所要する計算時間は、変数の更新ベクトルを求めるために解く一次方程式の計算に集中する。本実験のように、変数、制約式数がともに一万を越える計算例では一次方程式の解法に費す時間は内点法による計算時間の 95% 近くを占める。この一次方程式の左辺行列は

$$\begin{bmatrix} X^{-1}Z & -A^t \\ -A & 0 \end{bmatrix} \tag{18}$$

という特徴的な形を持っている[28] 。これによって行列のどこに非零が現れるかという構造もある程度限定される。シミュレーション型モデルに対して (18) 式の A 相当部分の非零な要素数は表 5 のようになり、全要素に対する非零要素数の比率が非常に小さいことがわかる。

表 5：多期間確率計画モデルの非零要素数

制約式の行列要素の総数	$\{(n+I)T+1\}(TI+2)$
非零要素の総数	$I\{(2T-1)(n+1)+2\}+2n+1$

図 7に示すように、経路数が増加するにつれて非零要素数はほぼ比例して増加するが、非零要素の比率はほぼ経路数の逆数に比例して減少する。これを行列の疎行列性と呼び、この性質を最大限に利用することが高速な一次方程式の解法、ひいては高速な内点法の解法を実現する上で最も重要となる。

本項では前項の計算実験に用いた内点法の実装上の技法、特に一次方程式解法の技法を説明する。

[28] (18) 式の X は決定変数 x_i を対角要素に持つ行列、Z は線形計画問題 ((17) 式) の最適性条件を表すときに必要な変数を対角要素に持つ行列である。以降の議論では、特に必要ないので、詳しくは省略する。

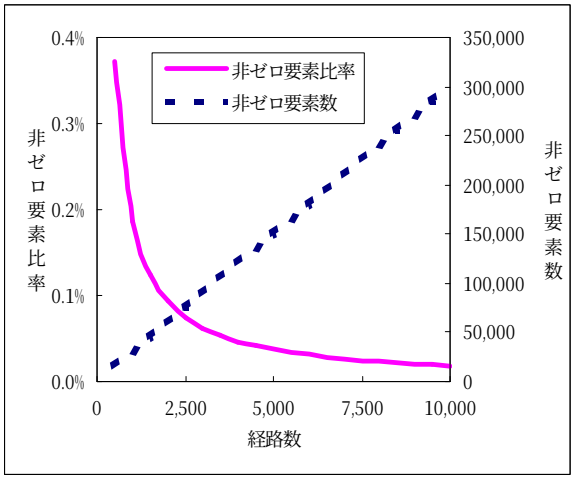

図 7：経路数と非ゼロ要素の関係 $(n = 3, T = 4)$

4.3.1　一次方程式の構造に特化した前処理の実行

(18) **式の非零の行列要素が現れる場所は制約式の係数行列** A **の非零パターンに依存するため、解法中不変である。そのため内点法の算法を開始する前に行列** A **の非零パターンが決定した段階で、非零パターンに特化したデータ構造を作成し、後の計算を効率化する。一次方程式の前処理の詳細は以下の通りである。**

① **一次方程式の変換による係数行列の構造の変換**

② **制約式、変数の番号付けの変更による係数行列の分解演算の省コスト化**

③ **係数行列のピボット選択入りの分解による係数行列の分解演算の安定化**

④ **係数行列の分解によって新たに現れる非零要素の場所の特定**

⑤ **係数行列の分解後の形について特徴的な構造の検出による分解演算の高速化**

一般に、一次方程式の前処理の計算コストは前処理によってもたらされる高速化に対し、引き合うものとなる。なぜなら一次方程式の解法は、通常内点法の解法全体で 20 **回から** 30 **回程度繰り返されるためである。前処理を行わない場合の一次方程式の解法時間を** T_0 **、前処理を行った場合の一次方程式の解法時間を** T_1 **、前処理時間を** T_p **、内点法の解法全体で必要な一次方程式の解法の回数を** m **とおくと、具体的には**

$$(T_0 - T_1) \cdot m \gg T_p \tag{19}$$

になることが前処理の計算コストが引き合うための条件である。5 **つある前処理の演算に所要する時間は元の行列の一次方程式の解法時間** (T_0) **を越えない。これは、各前処理演算が行列の一次方程式の解法と同じオーダーの計算量を所要することから言える。そのため、**

$$T_p \leq 5 \cdot T_0 \tag{20}$$

が成りたつ。したがって、m **が** 20〜30 **程度の場合には明らかに** (19) **式が成り立つ。**

4.3.2 一次方程式の変換 (Normal Equation Form)

(18) 式を係数行列とする一次方程式は式変形により

$$ADA^t \tag{21}$$

を係数行列とする一次方程式に変換することができる。ここで D は $Z^{-1}X$ である。この形を Normal Equation Form と呼ぶ。これは Adler *et al.*[1] によって提案された手法であり、以下の理由により効率化に貢献する。

① 行列サイズは制約式の数となり縮小される。

② 扱うべき係数行列である (21) 式 が正定値行列となる。

③ (21) 式の要素の計算時に必要な演算の一部を前処理で行っておくことができる。

ただし、シミュレーション型モデルのように A に密な列が含まれている場合、A が疎行列でも、(21) 式は密行列となり、この変形が逆に計算効率を下げてしまう。この場合の結果が、4.2項の表3である。この対策として、

① Column 分割 (Vanderbei[29])

② Shur Complement の利用 (Andersen[2])

③ Augmented System Approach(Maros and Mészáros[15])

のような手法が提案されているが、本計算実験では Augmented System Approach を用いている。Augmented System Approach は

① 密な列を特定して、列の番号付けの変更によって密な列を末尾に移動する

② 密な列の手前までに対して (21) 式に相当する変換を行う

という手法である。図8にその概念図を示す。

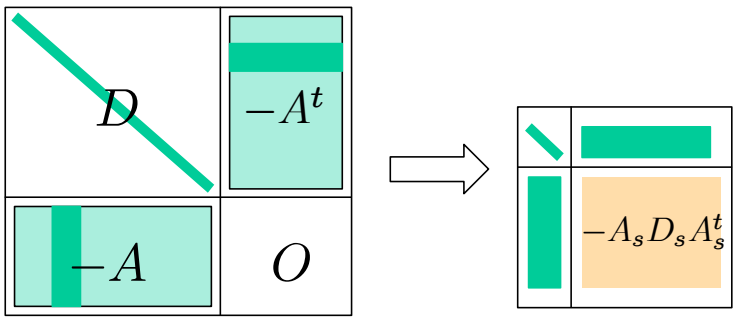

図 8： Augmented System Approach 概念図

4.3.3 変数番号の付け替え

前項の Augmented System Approach で変換された一次方程式の係数行列の列、行の並べ替えを行うと、行列の分解を行う際の非零要素数を削減することができる。並べ替えは行列の対称性を保つため、列と行を同時に入れ替える方法を採用する。並べ換え操作はオーダリングと呼ばれる。対称行列のオーダリングがもたらす効果の概念図を図9に示す。

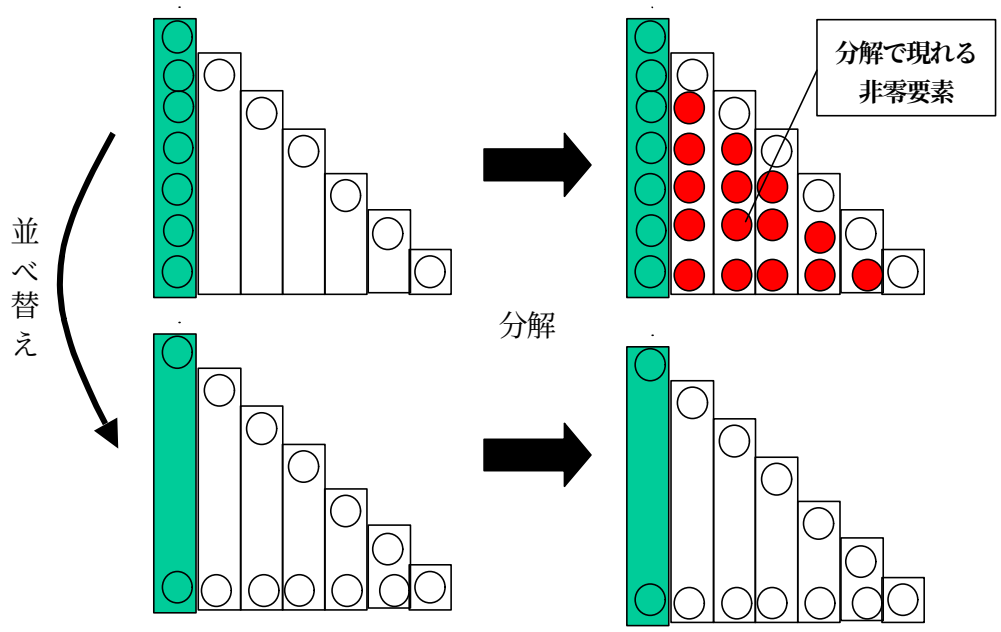

図 9：re-ordering の効果

表 6は、ランダムな非零要素構造を持つ二次計画問題 (変数総数:2,000, 制約式総数:1,000) について、オーダリングの有無による、非零要素の数と分解時間を示したものである。オーダリングによって、非零要素が 1/8 程度に押えられ、計算時間は 1/20 程度になっていることがわかる。

表 6：オーダリングの効果 (二次計画問題)

オーダリング	なし	あり
計算時間	39 秒	2 秒
非零要素数	116,860	14,242

(**使用計算機**：Pentium450MHz, **メモリ** 384M **バイト**)

オーダリングを発生する手法として以下の手法が知られている。

① Minimum Degree (Markowitz[14])：**非零要素数最小の** pivot **を順に選ぶ。**

② Minimum Local Fill (Lustig *et al.*[13])：**引き起こす** fill-in **最小の** pivot **を順に選ぶ。**

③ Inexact Minimum Local Fill (Mészáros[18])：Minimum Degree **のタイブレークに** Minimum Local Fill **を利用する。**

これらのアルゴリズムはすべてヒューリスティクスであり、その優劣は実装手段や問題の性質に拠る。複雑なアルゴリズムは良いオーダリングを発生するが、実行に時間を要する。その後の演算の削減への寄与でそれが相殺できるかどうかは、内点法全体に所要する行列の分解の回数などにも依存する。表 7は、一般の線形計画問題 (変数総数:462, 制約式総数:2,075) について Minimum Degree (MMD) **と** Inexact Minimum Local Fill (MLF) **を比較したものである。オーダリング発生時間では、**Minimum Degree **が有利だが、**Inexact Minumum Local Fill **は非零要素数の少ない行列を生成するので、全体の計算時間では勝っている。**

表 7 : MMD と MLF の計算時間比較 (一般の線形計画問題)

	MMD	MLF
オーダリング発生時間	0.6 **秒**	1.2 **秒**
非零要素数	465,652	406,719
計算時間	161 **秒**	128 **秒**

(**使用計算機** : Pentium450MHz, **メモリ** 384M **バイト**)

シミュレーション型モデル (**パス数** 5,000, **変数総数**:20,023, **制約式総数**:20,063) [29]**の場合の比較は表** 8**のようになり、行列の非零要素数は殆ど変化しないが、オーダリング生成時間が若干異なるため、**Inexact Minimum Local Fill **を採用している。行列の分解コストを支配する非零要素数が同じであるにもかかわらずオーダリング生成時間の差である** 2 **秒が全体の計算時間にそのまま反映していないが、この理由については、**4.3.5**項で述べる。**

表 8 : MMD と MLF の計算時間比較 (シミュレーション型モデル)

	MMD	MLF
オーダリング発生時間	59 **秒**	57 **秒**
非零要素数	275,419	275,416
計算時間	138 **秒**	138 **秒**

(**使用計算機** : Pentium450MHz, **メモリ** 384M **バイト**)

4.3.4 係数行列のピボット選択入りの分解

Augmented System Approach **を用いた場合、変換後の行列について正定値性は一般に保証できない。そのため、分解の際のピボットの選択に配慮が必要で、その際には行列要素の値を参照する必要がある。しかし、行列要素の参照を常に行っていると計算コストがかかるので、ここでは便法として、以下の方法を用いる** (**破綻が起きた場合にはその時点でピボット選択付きの分解をやり直す**)。

① **ピボット選択付きの分解は、内点法の最初のステップの分解についてのみ行う。**

② **以降は最初のステップの分解の際のそのピボット選択順を記憶して使う。**

内点法のステップ間で解く一次方程式の性質はあまり大きく変化しないので、②は有効であると予想される。実際、シミュレーション型モデルを含め、多くの線形計画問題の実験で最初の分解のピボット列を最後まで用いても破綻は起きないことが実験的に確かめられている。

本計算実験ではピボット選択付きの分解アルゴリズムとして、bunch-palett+multifrontal **分解** (Duff *et al.*[4]) **を用いている。これはピボットをスカラでなく、**2×2 **行列とする分解で、算法全体は** multifrontal **法** (sub-matrix cholesky **の一種** [3, 12]) **に基づいて進行するものである。ピボット選択は行列全体でなく、その一部を抜き出した** sub-matrix **内部で行われるため演算効率上有利**

[29] **この実験に利用したモデルでは、若干の投資制約に相当する制約式が加わっているので、変数、制約式の総数はこの論文に解説している基本型と若干異なっているが本質的な構造は変化していない。**

である。行列全体に対してピボッティングを行う方法は、疎行列のデータ構造の書き換えとなるので効率上不利である。

4.3.5　密行列演算への変換 (supernode の利用)

行列解法アルゴリズムの性質上、係数行列の分解後の構造は同一のパターンの繰り返しが現れやすい。そのようなものをまとめて supernode(George and Liu[5]) と呼び、密行列として扱うようにデータ構造を工夫する。supernode 内は密行列であることが前提とされるので、この処置を行うことによって行列を蓄えるのに必要な index 情報は減少する。また、行列の分解アルゴリズムとしてこの supernode 構造を意識したものを用いることによって、大規模疎行列の演算を密行列の演算に帰着させることができる。密行列の演算は一般に間接参照を含まないため、疎行列の演算に比べて高速に動作する実装が可能である。本計算実験では疎行列の分解を

- 密行列 ← 密行列×ベクトル
- 密行列の分解

という演算に帰着させて、BLAS[10] として知られる高速なサブルーチンパッケージを利用する。

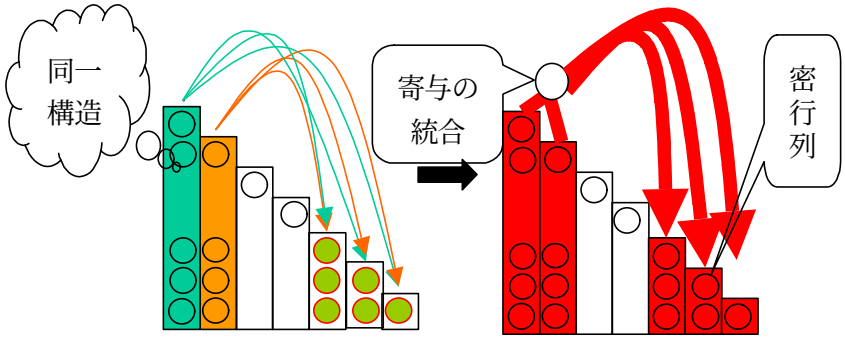

図 10：supernode 構造を利用した効率化

次は大規模線形計画と多期間確率計画問題で、密行列演算の高速化を行った場合と行わない場合で、全体の計算時間への影響を調べたものである。残念ながらシミュレーション型モデル (パス数 5,000, 変数総数:20,023, 制約式総数:20,063) での寄与はあまり大きくはないが、より規模が大きくなった場合に効果がより顕著に現れることが期待される。

表 9：supernode 化による高速化

	なし	あり
LP1(変数総数:4,883)	165 秒	161 秒
LP2(変数総数:12,230)	863 秒	743 秒
シミュレーション型モデル	140 秒	137 秒

(使用計算機：Pentium450MHz, メモリ 384M バイト)

4.3.3項の表 8に述べる計算結果において、Minimum Degree オーダリングと Minimum Local Fill オーダリングがほぼ同数の非零要素を持つ行列を発生しており、Minimum Local Fill のオー

ダリング生成時間が 2 秒少ないのに、全体の計算時間が変わらないのは、supernode 構造が異なるためであると考えられる。表10は前掲の表 8 に、行列を表現するための index 情報量と 1 回あたり行列分解所要時間を加えたものである。ここからわかるように index 情報量は Minimum Degree で生成された行列の方が少ない。このことから Minimum Degree で生成された行列から得られた構造の方が、密行列となっている部分が多いことが推定される。そのため 1 回あたりの分解時間が若干加速し、その加速分がオーダリング生成時間を相殺していると考えられる。

表 10：MMD と MLF の計算時間比較 (シミュレーション型モデル:再掲)

	MMD	MLF
オーダリング発生時間	59 **秒**	57 **秒**
非零要素数	275419	275416
計算時間	138 **秒**	138 **秒**
index **情報量**	275275	275295
1 回あたり分解時間	0.42 **秒**	0.43 **秒**

(**使用計算機**：Pentium450MHz, **メモリ** 384M **バイト**)

4.3.6 行列の分解算法の選択

行列の分解算法については

① row-wise Cholesky

② column Cholesky(left-looking)(Ng and Peyton[23])

③ multifrontal

④ right-looking(Rothberg and Gupta[24])

などの算法が知られている [3]。これらは数学的には等価だが、実装した場合、キャッシュメモリのヒット効率などを考慮すると効率に違いが生じる。本計算実験では、4.3.5項の supernode を考慮した場合のキャッシュメモリのヒット効率の点から right-looking を採用している。

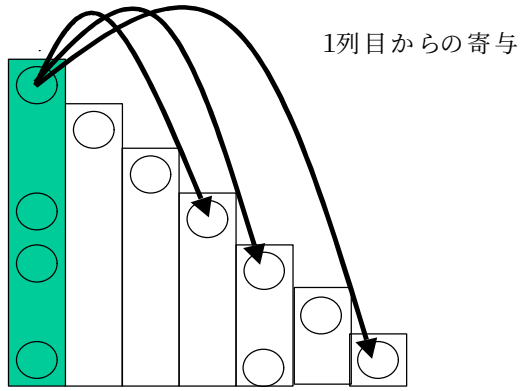

図 11： right-looking **アルゴリズムの概念図**

5　結論と今後の課題

本研究では、多期間ポートフォリオ最適化問題に対する数理計画のモデリング技術と実装技術の重要性を中心に議論した。多期間確率計画モデルの定番であるシナリオ・ツリー型モデルに対しては 1990 **年代において様々なモデルや高速に問題を解くためのアルゴリズムが提案されてきた。一方、本論文でとりあげたシミュレーション型モデルで必要なサンプル・パス (シミュレーション経路) は、資産価格変動 (収益率) の振る舞いに対して複雑なモデルを想定したとしてもモンテカルロ・シミュレーションによって生成することが比較的容易である。モンテカルロ・シミュレーションを利用したリスク評価システムによってポートフォリオ管理を行っている場合も多く** [19]**、これらのリスク評価システムに本モデルによるリスク制御を導入することも可能である。この点から考えると、シミュレーション型モデルは極めて有望なモデル化の方法である。しかも、線形 (凸) 計画問題として定式化を行うことにより、大域的最適解の導出を保証している。**

しかし、資産数と期間数に依存する次元数が多くなると、より多くのシミュレーション経路数が必要となり、計算精度向上のためには高速な計算アルゴリズムによる最適解導出は必須条件である。本研究では、現在開発されている様々な技法を比較・検討し、シミュレーション型モデルに最も適切な方法を用いた実装を行った。シミュレーション型モデルは提案されたばかりであり、モデルのバリエーションやアルゴリズムなど開発の余地も多い。シミュレーション型モデルを発展させたシミュレーション/ツリー混合型モデル [8] **もシミュレーション型モデルと同様にナイーブな実装を行うと計算時間を多く必要とする。しかし、本論文で示した実装技術を用いることによって、高速化できることが確かめられている。今後は、他の様々なタイプの多期間ポートフォリオ最適化モデルに対しても研究が必要であろう。計算機の高速化・低価格化に伴い、金融・投資分野における実際の問題に対し、数理計画を含む** OR **手法の計算可能性が高くなり、その融合はますます重要になってきている。**

参考文献

[1] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga, Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm, *ORSA Journal on Computing*, **1** (No.2, 1989), pp.84–106.

[2] K. D. Andersen, A Modified Schur Complement Method for Handling Dense Columns in Interior Point Methods for Linear Programming, Technical report, Dept. of Math. and Computer Sci., Odense University, 1994. Submitted to ACM Trans. On Mathematical Software.

[3] I. S. Duff, A. M. Erisman, and J. K. Reid, Direct Methods for Sparse Matrices, Oxford University Press, New York, 1989.

[4] I. S. Duff, N. I. M. Gould, J. K. Reid, J. A. Scott, and K. Turner, The Factorization of Sparse Symmetric Indefinite Matrices, *IMA J. Numer.Anal.*, **11** (1991), pp.181–204.

[5] A. George and J. W.-H. Liu, Computiong Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[6] **枇々木規雄**, **金融工学と最適化**, **朝倉書店**, 2001.

[7] **枇々木規雄**, **戦略的資産配分問題に対する多期間確率計画モデル**, *Journal of Operations Research Society of Japan*, **44**, No.2(2001), pp.169-193.

[8] **枇々木規雄：最適資産配分問題に対するシミュレーション/ツリー混合型多期間確率計画モデル**, **高橋一編**, **ジャフィー・ジャーナル** [2001] **金融工学の新展開**, 2001 **年** 6 **月**, pp.89-119.

[9] **今野浩**, **理財工学** II — **数理計画法による資産運用最適化** —, **日科技連**, 1998.

[10] C.L. Lawson, R.J. Hanson, D. Kincaid, and F.T. Krogh, Basic Linear Algebra Subprograms for FORTRAN usage, *ACM Transactions on Mathematical Software*, **5** (1979), pp.308–323.

[11] J. W. H. Liu, A Compact Row Storage Scheme for Cholesky Factors Using Elimination Trees, *ACM Transactions on Mathematical Software*, **12** (No.2, June 1986), pp.127–148.

[12] J. W. H. Liu, The Multifrontal Method and Paging in Sparse Cholesky Factorization, *ACM Transactions on Mathematical Software*, **15** (No.4,December 1989), pp.301–325.

[13] I. J. Lustig, R. E. Marsten, and D.F.Shanno, Interior Point Methods for Linear Programming: Computational state of the art, *ORSA Journal on Computing*, **6**(No.1, 1994), pp.1–15.

[14] H. M. Markowitz. The Elminiation Form of the Inverse and Its Application to Linear Programming, *Management Science*, **3**(1957), pp.255–269.

[15] I. Maros and C. Mészáros, The Role of the Augmented System in Interior Point Methods, *European Journal of Operational Reserach*, **107**(1998), pp.720–736.

[16] R.C. Merton, Lifetime Portfolio Selection under Uncertainty: The Continuous-Time Case, *The Review of Econmics and Statistics*, **51** (1969), pp.247–257.

[17] E. Messina and G. Mitra, Modelling and Analysis of Multistage Stochastic Programming Problem: A Software Environment, *European Journal of Operational Research*, **101** (1997), pp.343–359.

[18] Cs. Mészáros, The "Inexact" Minimum Local Fill-in Ordering Algorithm, Working paper WP 95-7, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1995.

[19] J.P. Morgan, RiskMetrics Technical Document (fourth edition), 1996.

[20] J.M. Mulvey and W.T. Ziemba, Asset and Liability Allocation in a Global Environment, Chapter 15 in *"Handbooks in OR & MS, Vol.9", edited by R.Jarrow et al., 1995.*
(**翻訳：枇々木規雄：グローバル環境における資産負債配分**, **第**15**章**, **今野浩**, **古川浩一編著**, **ファイナンスハンドブック**, 1997.)

[21] J.M. Mulvey and W.T. Ziemba, Asset and Liability Management Systems for Long-Term Investors: Discussion of the Issues, Chapter 1 in *"Worldwide Asset and Liability Modeling", edited by W.T. Ziemba and J.M. Mulvey, 1998.*

[22] P.A. Samuelson, Lifetime Portfolio Selection by Dynamic Stochastic Programming, *The Review of Econmics and Statistics*, **51** (1969), pp.239–246.

[23] E. G. Ng and B. W. Peyton, Block Sparse Cholesky Algorithms on Advanced Uniprocessor Computers, *SIAM Journal of Scientific Computing*, **14** (No.5, September 1993), pp.1034–1056.

[24] E. Rothberg and A. Gupta, Efficient Sparse Matrix Factorization on High-Performance Workstation-Exploiting the Memory Hierarcy, *ACM Transactions on Mathematical Software*, **17** (No.3, September 1991), pp.313–334.

[25] (**株**)**数理システム**, NUOPT **マニュアル**, 2000.

[26] (**株**)**数理システム**, SIMPLE **マニュアル**, 2000.

[27] **竹原均**, **ポートフォリオの最適化**, **朝倉書店**, 1997.

[28] **田辺隆人**, **金融工学と数理計画法**, **日本金融・証券計量・工学学会** 1999**年冬季大会予稿集**, pp. 56–65.

[29] R. J. Vanderbei, Splitting Dense Columns in Sparse Linear Systems, *Linear Algebra Appl.*, **152**(1991), pp.107–117.