非線形計画法アルゴリズムの実装と応用

（株）数理システム　田辺隆人

非線形計画法（NLP）の適用について実装上の観点からまとめた．まずアルゴリズムを概括し，実装においては線形計画法（LP）の高速化技術がNLPの場合にも有効であることを示す．つづいて複雑なNLPにいおいては関数値とその微係数の計算手段が計算効率上重要となることを示し，その実装としてモデリング言語と連動した自動微分法を紹介する．最後にNLPが用いられる具体的な応用分野について述べる．

1．アルゴリズム

NLP求解アルゴリズムで現在主流なものとしては，逐次二次最適化法（SQP）と内点法の二つが知られている．SQPの反復は現在の反復点のまわりで目的関数・制約式をそれぞれ二次・一次関数で近似した二次計画（QP）問題を解くことであり，通常の実装ではQPの解法として厳密解法である有効制約法が用いられる．有効制約法は解においてactiveな（等式として満たされる）制約式の集合を「ピボッティング」と呼ばれる操作によって入れ換えながら解を探索するという性質上，activeな制約の集合があらかじめ正確に得られている場合には求解が高速化される．SQPの各反復では性質の似通ったQPを連続して解くことになるが，各反復でactiveとなる不等式制約の集合を次回の反復で利用することによって全体を高速化することが可能である．有効制約法はLP用の単体法の拡張と考えられるため，基底の更新・プライシング・基底行列の分解などの操作において単体法の実装技術をそのまま活かすことができる．そのためいくつかの手続きを共用する形で単体法と有効制約法を同一のプログラム内に共存させることは自然に可能である．次はポートフォリオ選択問題においてリスク尺度（ポートフォリオの評価基準）を分散で行った場合と絶対偏差[2]で行った場合の実行速度の比較である．前者は目的関数が二次関数となるのでQPに，後者は絶対値となるのでLPになるが，変数sの絶対値の最小化を定式化するにあたって

$$\text{最小化}|s| \quad \Rightarrow \quad \text{最小化}\ s_1+s_2,\ \text{ただし}\ s=s_1+s_2,\ s_1,s_2\geq 0 \qquad (1)$$

という書き換えを行っているので変数の数は異なる．以下では実行速度はピボットの数が少ないQPの方がむしろ高速という結果になっている．これは類似した性質を持つLP・QP問題ならば，QPであるがために特段実行速度が低下することはないということを示している．

	QP＋有効制約法				LP＋単体法			
銘柄数	変数	制約	ピボット	計算時間	変数	制約	ピボット	計算時間
1000	1060	62	111	0.25秒	1120	62	453	0.49秒
5000	5060	62	72	0.91秒	5120	62	332	2.09秒
10000	10060	62	69	1.86秒	10120	62	303	3.60秒

（利用マシン Pentium1.5GMHz +1G バイトメモリ　ソフトウエア：NUOPT5.2.1）

例えば[1]に紹介されている内点法の算法では特に大規模問題においてLP・NLP（QP）いずれの場合にも次の構造の対称不定値行列を左辺とする一次方程式解法に大部分の計算時間が消費される．

$$\begin{bmatrix} G+D & -A^t \\ -A & 0 \end{bmatrix} \qquad (2)$$

ただし，問題は標準形：

$$\text{変数}：x\in\mathbf{R}^n \quad \text{最小化}\ f(x)$$
$$\text{制約}：g(x)=0,\ x\geq 0,\ g(x)\in\mathbf{R}^m \qquad (3)$$

に変形されているとし，Aは制約式のヤコビ行列（$A\equiv\nabla g(x)\in\mathbf{R}^{m\times n}$）$G\in\mathbf{R}^{n\times n}$はラグランジュ関

数（$L \equiv f(x) - A^t y - x^t z$）のヘッセ行列あるいはその近似，行列$D \in \mathbf{R}^{n \times n}$は主・双対変数の値から生成される対角行列である．

一万変数を超えるサイズの問題で一次方程式解法は全求解時間の約75%以上の時間を占めることから，一次方程式解法が内点法の実装技術の最も重要な部分と言える．LPの場合にはGが零であることからこの行列の左上部分は対角行列となる．その場合には一次方程式の左辺を

$$\begin{bmatrix} D & -A^t \\ -A & 0 \end{bmatrix} \Rightarrow \left[-AD^{-1}A^t \right] \qquad (4)$$

と変換することができる[3]．この変形は特にAの行の数（制約式の数）が少ない場合の行列サイズの削減による高速化や，正定値であることを利用した行列分解手法の簡略化などのメリットがある．反面，Aが非零要素を多く含む列（dense column）を持つときには行列の疎行列性を破壊する結果を招くので，（1）の要素を並べ替えてから（2）の変形を部分的に行う方法Augumented System Approach[4]が提案された．

$$\begin{bmatrix} D_s & 0 & A_s^t \\ 0 & D_d & A_d^t \\ A_s & A_d & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} D_d & A_d^t \\ A_d & -A_s D_s^{-1} A_s \end{bmatrix} \qquad (5)$$

ここで，A_dはAの各列のうち，非零要素の多いもののみから構成された行列，D_dはそれに対応する対角要素である．A_s, D_sはその残りの列に対応する．この方法が$G \neq 0$である一般のNLPの場合に拡張可能であることは明らかである．例えばD_sに対応する部分のG（G_sと書く）が対角行列（あるいは零行列）となり，$D_s + G_s$の対角要素が零にならないことが保証できる場合には$D_s \rightarrow D_s + G_s$とすることでこの変形がそのまま利用できる．実際のNLPでは多くの線形制約を含み，また，スラック変数など制約式の線形項にしか現れない変数も多いことから，この方法は有効であることが確かめられている．

この方法の難しさは変形の結果現れた（5）の右辺の正定値性が保証できないため，行列の分解にピボット選択が必要になる点である．LPの場合にはAがrow-full rankであるなどの適当な仮定の下で（5）の右辺行列の分解時に零のピボットが現れない（分解が破綻しない）ことを保証できる[3]が，その保証が成たたないNLPの場合にはピボット選択を行うBunch-Palettet分解を行う必要がある．ピボット選択を常に行うと計算量が増大するので，内点法の各反復では，前回の反復で得られたピボット選択順序を再利用して計算効率を上げる方法が考えられる．実際内点法によって解かれる行列は急激に性質が変化しないのでこの方策はおおむね成功し，実際に分解が破綻してピボット選択をやり直す必要は一つの問題（反復回数：２０～５０回前後）に対しておおむね１回以下であることが実験的に確かめられている．ピボット列が固定されている場合の一般構造を持つ対称行列の分解の効率化については例えばLPの内点法による解法[5]や構造解析，シミュレーションの文脈でノウハウが蓄積されている．本稿ではその詳細を述べることはできないが，一般に効果的なものを挙げると

1. 分解後の行列の非零要素数減少のためのオーダリング
2. Supernodal アルゴリズムとノードamalgamation
3. BLAS3 カーネルの利用によるマシン特性の利用

のようになる[10]．特に3．については高品質で安価なBLAS3カーネル[6]がWindowsを含めた多くの環境においてフリーで提供されるようになったために計算速度の向上に大きく貢献している．次は大規模なLP，NLP問題について2．および3．の効果を実測したものであり，これらが内点法のLP，NLP問題の解法に等しく貢献していることがわかる．

問題種別	変数	制約	2，3あり	2，3なし
LP（大規模ポートフォリオ）	12230	6072	75秒	150秒
NLP（非線形ネットワーク）	112769	44757	2102秒	8828秒

（利用マシンPentium1.5GMHz+1Gバイトメモリ　ソフトウエア：NUOPT5.2.1）

2. モデリング

式の次数が限られている LP や QP の場合は式を制約式の係数行列やヘッセ行列という形で汎用的に表現することができるが，NLP は式の次数や関数形に制限がないので，行列に代わるものとして計算グラフと呼ばれる新しい形式による表現を考える．計算グラフは式（中間変数）の計算の各ステップを初等的な演算（四則演算や初等関数）に分割，各ステップによって計算される値をグラフのノード，ノード間の依存関係をグラフの有向アークで表現したものである．問題全体を一つの計算グラフとして考えた場合，制約式や目的関数（以下総称して「関数」）からその依存しているノードを辿ってゆくと最後には変数に行き着く．NLP の複雑さは問題を記述した計算グラフのノード数として計測することができる．逆に変数からその寄与しているノードを逆に辿ると制約式に行き着く．自動微分算法を用いるとこのような計算グラフの横断を行うことによって任意の関数，任意の変数に対する微係数を求めることができる．

計算グラフの作成は式の記述に等しい．行列の場合と同様に，大規模で複雑な計算グラフを手動で生成するのは一般に困難なので，モデリング言語などの問題入力支援ツールが必要となる．例えばあるプラント最適化問題では変数の数 440，制約式の数 294 と比較的小規模であるのにも関わらずモデル記述には約一万行を必要とし，そこからノード数 62574 個の計算グラフが生成される．例えば汎用のモデリングツールである SIMPLE[7]はモデルに現れる繰り返し構造をまとめて記述する文法を持ち，LP，QP の記述に対しては行列を，NLP の記述に対しては計算グラフを生成，自動微分アルゴリズムによって非線形関数の微係数を提供する．非線形最適化では，この微係数の計算時間が実際の計算に影響する．LP や QP では制約式の係数行列（ヤコビ行列に相当）や目的関数のヘッセ行列は不変であるのに対して，非線形関数の場合にはそれらは変数の値に依存して変化するので，アルゴリズムが変数値を更新するたびに微係数の計算が必要となるためである．先のプラント最適化問題では最適化全体に必要な時間 130 秒のうち，微係数の計算に全体の 114 秒（88%）の時間を消費している．

微係数の計算手法はアルゴリズムの選択にも影響を及ぼす．次は信用リスクを含む債券の価格付けの際に解かれる推移確率行列推定問題で，与えられた非対称行列 Q_{year} の５２乗根となる行列で特定の条件を満たすものを求める問題である[8]．

$$\text{変数：} Q \in \mathbf{R}^{18\times 18} \quad \text{最小化} \left\| Q_{year} - Q^{52} \right\| \text{：フロベニウスノルム}$$

$$\text{制約：} \sum_j Q_{ij} = 1,\ Q_{ij} \geq 0 \qquad (6)$$

次は上記の問題を二つのアルゴリズム（直線探索法・信頼領域法）で求解した場合の計算時間である．この問題の場合にも先のプラント最適化問題と同様に，変数・制約式の数に比較して計算グラフのノード数が 46901 個と規模が大きなことから目的関数の微係数の計算に多くの時間を所要する．直線探索法は 1 階微係数のみを必要とするのに対して信頼領域法は 2 階微係数をあわせて必要とする．2 階微係数の計算には多くの時間を所要するので，1 回あたりの反復では直線探索法が高速だが，信頼領域法は 2 階微係数を用いているので，直線探索法に比べて収束が速く，全体の計算時間では信頼領域法が有利となっている．

		直線探索法		信頼領域法	
変数	制約	反復	計算時間	反復	計算時間
324	18	337	2125 秒	78	1616 秒

(利用マシン Pentium1.5GMHz +1G バイトメモリ　ソフトウエア：NUOPT5.2.1)

次も類似の格付け推移確率行列の期間構造の推定問題の例(計算グラフノード数 65316 個) であるが，類似の状況が発生している．

		直線探索法		信頼領域法	
変数	制約	反復	計算時間	反復	計算時間
72	841	389	875 秒	63	393 秒

(利用マシン Pentium1.5GMHz +1G バイトメモリ　ソフトウエア：NUOPT5.2.1)

最初に述べたプラント最適化の問題と同様に上記の二例でも信頼領域法を適用した場合には自動微分演算には全体の 90%以上を所要しており，総じて変数や制約式の数が比較的少ない複雑な非線形最適化の高速化には自動微分の高速化が重要と言える．

モデリング言語や自動微分の実行効率は実装の方法に大きく依存する．期間構造の推定問題においては，自動微分法の実装を変数が行列の形をしているという構造を活かしてチューニングしたところ，計算効率は約 40 倍になったという報告[9]がある．一般に最適化問題の記述には添字付けを用いた繰り返し構造が現れることから計算グラフや自動微分算法そのものにもこの構造を取り入れることによって計算効率を上げる試みを試行中である[11].

3. NLP の具体例

NLP が現れる局面としては例えば以下がある．

1. 混合問題の一般化
2. 金融工学の応用例
3. プラント解析

は混合前の物質の物性と混合後の物性の非線形性より，2. は金利計算に現れる log/exp 関数・分布関数の関数形より，3. は物性を記述する方程式の非線形性より NLP でのモデリングが必要となる．特に 3. では機器の運転を表す０－１変数をあわせて含むことにより，組み合わせ問題としての性質も備えるのでさらに複雑である．これらの問題に対する NLP の具体的な適用例については当日会場にて示す．

参考文献

[1] H. Yabe, H. Yamashita, and T. Tanabe, A globally and superlinearly convergent primal-dual interior point trust region method for large scale constrained optimization, Technical Report, Mathematical Systems Inc., Tokyo, Japan, July 1997 (revised July 1998).

[2] 今野　浩, 理財工学 I, 日科技連 1995

[3] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga. Data structures and programming techniques for the implementation of Karmarkar' s algorithm. ORSA J. on Comput., 1(2):84-106, 1989.

[4] I. Maros and Cs. Meszaros. The role of the augmented system in interior point methods. Technical Report TR/06/95, Brunel University, Department of Mathematics and Statistics, London, 1995.

[5] E. D. Andersen, J. Gondzio, Cs. Meszaros, and X. Xu, Implementation of interior point methods for large scale linear programs. In T. Terlaky, editor, Interior point methods of mathematicall programming, pages 189-252 Kluwer Academic Publishers, 1996.

[6] R. C. Whaley, A. Petitet, J. J. Dongarra, Automated Empirical Optimization of Software and the ATLAS project, Technical report, University of Tennessee, Knoxville, TN, Department of Computer Science, Univ. of TN, Knoxville, TN 37996, April 2000.

[7] 山下浩, 田辺隆人, 逸見宣博, 数理科学のためのモデリング言語 SIMPLE, 研究報告 「数理モデル化と問題解決」 アブストラクト No. 004 - 005
(http://www.ipsj.or.jp/members/SIGNotes/Jpn/23/1995/004/article005.html)

[8] 楠岡成雄・青沼君明・中川秀敏, クレジット・リスクモデル, きんざい 2001

[9] 青沼君明, 田辺隆人, An Estimation for The Term Structure of Yield Spread, 日本オペレーションズ・リサーチ学会 金融工学研究部会 2001 (http://www.socs.waseda.ac.jp/or-finance/2001.html)

[10] 田辺隆人, 対称行列の分解の高速化, テクニカルレポート, (株) 数理システム 2001

[11] 田辺隆人, 添字付け記法による自動微分高速化, (株) 数理システム 2002