解説

数理計画の理論と実装

非線形計画法アルゴリズムの実装と応用

田辺　隆人

1. はじめに

最適化問題：

最小化：$f(x)$

制　約：$g(x)=0,\ x\geq 0\quad x\in\mathbf{R}^n,\ g(x)\in\mathbf{R}^m$　(1)

において，目的関数$f(x)$，制約式$g(x)$が一般の非線形関数であるような問題を非線形計画（NLP）問題と呼ぶ．本稿ではこの求解アルゴリズムの実装とその応用分野について具体的に解説する．まずアルゴリズムと実装を概括し，線形計画法（LP）の実装技術がNLPに継承されていることを示す．次に内点法の実装においてもっとも重要な一次方程式解法部分の実装に着目，LPの場合との類似点および相違点について述べる．続いてNLPの応用に共通する話題として，モデリングおよび自動微分について解説する．最後に現実の問題においてNLPが利用される分野を具体的に解説する．

2. アルゴリズムと実装

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（Sequential Quadratic Programming：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：

最小化：x^tQx+c^tx

制　約：$Ax=b,\ x\geq 0$　(2)

を解くことに帰着される．したがって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，その場合の解法としては厳密解法である有効制約法が用いられることが多い．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．有効制約法による求解はactiveな制約のよい見積もりが得られる場合には高速化されるため，有効制約法をSQPの求解に用いる場合には，あるSQPの反復で解くQPに対するactiveな制約を，次の反復で解くQPのactiveな制約の見積もりとし，全体の求解の高速化を試みる．この事情は混合整数線形計画問題における分枝限定法の解法に単体法が利用されるのと類似している．

有効制約法は線形計画（LP）問題：

最小化：c^tx

制　約：$Ax=b,\ x\geq 0$　(3)

に対する単体法の拡張であり，単体法の実装技術である：

初期基底の作成（crash）

実行可能解の生成（phase-I）

基底の分解・更新

プライシング

などのノウハウをそのまま活かすことができる．

すなわち，

単体法 ⇒ 有効制約法 ⇒ SQP

という形で実装技術の継承が行われているといえる．実際の数値実験においても単体法と有効制約法のパフォーマンス上の差は比較的小さい．大規模QPを解くことの多い金融分野の例を見てみよう．

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

最小化$|s|$ ⇒ 最小化s_1+s_2,

ただし$s=s_1+s_2, s_1,\ s_2\geq 0$　(4)

という書き換えを行っているので変数の数は異なる．

表1の結果によると実行速度はピボッティングの数

たなべ　たかひと
㈱数理システム
〒160-0022 新宿区新宿2-4-3

表1　ポートフォリオ問題（LP，QPの実行結果）

	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 Pentium1.5GHz +1G バイトメモリ　ソフトウエア：NUOPT5.2.1，以下の計算例についてすべて同一）

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．このことから，例えばプラント運転計画や電力潮流計算[12]などにおいて，LPモデルをQP化して精度を向上するというアプローチが現実的であることがわかる．

もう一つの流れである内点法によるNLP解法についても同様の実装技術の継承が見られる．

内点法は最適性の必要条件であるKKT条件：

$$\begin{aligned} &g(x)=0,\\ &\nabla f-A^t y-z=0,\\ &XZe=0,\\ &x\geq 0,\quad z\geq 0 \end{aligned} \tag{5}$$

の後半の2式を

$$\begin{aligned} &XZe=\mu,\\ &x>0,\quad z>0 \end{aligned}$$

のように変形し（μはパラメータ），非線形方程式系と見て解く手法である．主変数x（双対変数z）のみを解く主（双対）内点法，両方を同時に解く主双対内点法，大域的収束性の保障のためにメリット関数を導入して直線探索・信頼領域を用いる方法，局所的収束性の向上のためにメリット関数の非単調ステップを許す方法，KKT条件のより高次の情報を利用する方法など，様々なバリエーションが知られており，その適用範囲（NLP，QP，LP）も異なる[13]．ただ，いずれも非線形方程式系を解くというアプローチにおいて共通している（例えばLPでもKKT条件は非線形方程式系となる）ため，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．具体的には1万変数を超えるサイズのQP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という実験例[14]がある．

$$\begin{bmatrix} G+D & -A^t \\ -A & 0 \end{bmatrix} \tag{6}$$

ここで，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（$L\equiv f(x)-g^t y-x^t z$）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・双対変数の値から構成される通常要素値が正の対角行列である．

表2は様々な数理計画問題に対する内点法の計算例である．QP，NLPでも規模が小さければ計算時間は短いことから計算時間と種別の間に目立った相関はないこと，計算時間は変数や制約式の数，すなわち式(6)のサイズによることがわかる．

次の項では式(6)の解法に着目し，LPのケースにおいて培われた実装技術がどのようにQP，NLPの解法に継承されているかを見る．

3．内点法の一次方程式解法

まず，式(6)の構造の特殊性に着目した効率化が考えられる．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

$$\begin{bmatrix} D & -A^t \\ -A & 0 \end{bmatrix} \Rightarrow [-AD^{-1}A^t] \tag{7}$$

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，行列が正定値であることを利用した行列分解手法が使えるなどのメリットがある．さらにLPの場合にはAが変化しないことを利用してデータ構造の効率化が可能である．反面，Aが非零要素を多く含む列（dense column）を持つとき，この変形は行列の疎行列性を破壊する（変換後の行列が密行列になる）結果を招くので，式(6)の行・列を並べ替えてから式(7)の変形を部分的に行う方法であるAugumented System Approach[4]が提案された．

$$\begin{bmatrix} D_s & 0 & -A_s^t \\ 0 & D_d & -A_d^t \\ -A_s & -A_d & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} D_d & -A_d^t \\ -A_d & -A_s D_s^{-1} A_s \end{bmatrix} \tag{8}$$

表2　内点法の実行例

問題名	種別	変数	制約式	反復	計算時間
資源計画	LP	76308	28129	44	37秒
ポートフォリオI	QP	6273	7874	48	3秒
ポートフォリオII	NLP	123	267	20	0.5秒
原料購入計画	NLP	4166	2172	20	103秒

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 D のうちで A_d に対応する要素を集めた対角行列である．A_s, D_s はそれぞれ残りの列に対応する．

この方法は $G\neq 0$ であるQP，NLPの場合に拡張可能である．もっとも単純なケースとして凸な非線形計画問題で G が対角行列であるケースでは，$G+D$ が対角要素で各要素が正なので式(7)あるいは式(8)の変形をそのまま使うことができる．G が一般の構造を持つ場合にも，G のうち，0または対角行列となり，かつ対応する A の列が密でない部分を G_s，それ以外の部分を G_D とするなら式(8)と同様に

$$\begin{bmatrix} G_s+D_s & 0 & -A_s^t \\ 0 & G_D+D_d & -A_d^t \\ -A_s & -A_d & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} G_D+D_d & -A_d^t \\ -A_d & -A_s(G_s+D_s)^{-1}A_s \end{bmatrix} \tag{9}$$

なる変形が可能となる．実際のNLPには線形制約にしか現れていない変数（G の対応する行，列が0），あるいは別の変数とのクロスタームが存在しない変数（G の対応する行，列の非対角要素が0）も多く現れるので，式(9)の変形によって行列の構造や性質がほとんどLPと同一となる場合も多い．すなわち，この方法を取ることにより，LPと性質が似通った問題は，LPと同様のパフォーマンスの享受が可能となる．ただ，一般には変形の結果現れた式(9)の右辺の正定値性が保証できないことに注意する必要がある．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式(8)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）ことを保証できるが[4]，その保証が成り立たないNLPの場合には常にピボット選択付きの例えばBunch-Parlett分解[15]を行う必要がある．Bunch-Parlett分解は（2×2）ピボットを用いることによって，ピボット選択を行っても行列の対称性を保つことができる点に特徴がある．

内点法ではその反復（20～50回前後）のそれぞれにおいて式(9)を解く必要が生じるが，ピボット選択を常に行うと計算量が増大するので，数値的条件の悪化が見られないかぎりピボット選択順序を再利用して計算効率を向上させる方法が考えられる．式(9)に示されている内点法の各反復によって解かれる行列は単体法と違い，常に構造（非零要素の場所）が同一で，数値的な性質も急激には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のやりなおしが生じるのは内点法の全反復中，おおむね1回以下で済むことが実験的に確かめられている．

このように内点法の大部分の反復ではピボット順を固定して行列を解くことを繰り返している場合には，さらなる高速化の工夫が可能である．具体的には，行列の構造とピボット順に依存した前処理を行い，行列の分解演算手順の減少や処理系（CPU，メモリ）の最適化機能の利用促進を図る工夫が重要となる．前者の一つが，オーダリング（列，行の再番号付け）であり，後者がSupernodal法として知られている．これらのノウハウは構造解析，流体解析，半導体のデバイス・プロセスシミュレーションの文脈で蓄積されたものが多い[10]．特にオーダリングの効果は劇的で，古くから知られるminimum-degreeアルゴリズム[20]を用いるのみでも一次方程式の解法時間が数十分の一に短縮できることで疎行列の分解には必須の技術となっている．オーダリングについてはいくつかのヒューリスティクスが提案されているが，最近はgraph-partitioningを用いた方法によって大規模問題に対するパフォーマンスの向上が報告されている[16]．

Supernodal法は疎行列の演算を密行列演算に帰着させるものである．そのため，密行列演算を処理系の最適化機能を活かして高速に実行するライブラリ（例えばBLASカーネル[17]）の利用が前提となる．以

表3　Supernodal法の効果

問題種別	変数	制約	Supernodal利用	Supernodal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

前，そのような実装は処理系のベンダによってのみ開発され，有料で配布されていたため汎用ソフトへの組み込みは難しかったが，近年には高品質で安価なBLASカーネルが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り，マシンの高速化と相まって，WindowsPCなどの一般的な環境でも高パフォーマンスな計算が可能となっている．例えば単純なコードで1200×1700の対称密行列の掛け算を行うのに，CPUがPentiumIV，1.9 GHzのWindowsマシンでは22秒を所要するが，BLASカーネルの実装の一つ，ATLAS[6]を利用すると，所要時間は1.7秒に高速化される．これは1 Gflopsを超えるパフォーマンスである．密行列ライブラリの性能向上によって，あえて疎行列性を無視して帰着できる密行列演算量を増やすことにより，パフォーマンス向上が観測される例もある．表3は大規模なLP, NLP問題についてSupernodal法の効果を実測したものである．Supernodal法の効果の度合いには差があり，問題が大規模で疎性が低い（密行列に近い）ほど有効に機能する．大規模LPのベンチマーク問題から得られた結果によるとSupernodal法の導入によって平均して30%程度の速度向上が得られることが実証されている．

4. モデリング

NLPの計算機上の作成や入出力について考えてみる．式の次数が限られているLP, QPの場合は問題全体を制約式の係数行列やヘッセ行列という定数行列で記述することができるので問題の取り扱いは定数行列の作成や入出力に帰着する（例えば古くから知られるMPS形式[21]はそのことを利用したLPの表現である）．対して一般のNLPの場合には，式の次数や関数形に制限がないため行列のみでは問題を表現することができない．このような場合に行列に代わる式の表現として計算グラフと呼ばれる新しい形式が知られている．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，

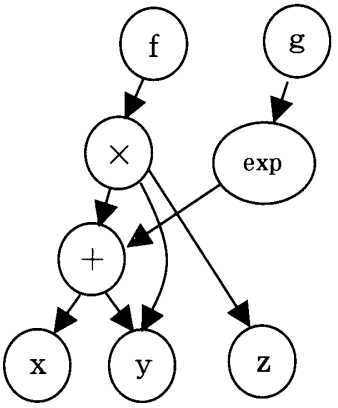

図1　計算グラフの実際例

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．図1は次の二つの式を定義した計算グラフの概念図である．

$$f=(x+y)\cdot y\cdot z$$
$$g=\exp(x+y)$$

この形式を利用するとNLP問題全体は一つの計算グラフとして表現される．NLPの複雑さは問題を記述する計算グラフのノード数として計測することができる．計算グラフのサイズによる複雑さの計測においては同一の関数の反復構造を除外することができることに注意されたい．例えば上の例で$(x+y)$に対応するコンポーネントがf, g両方の表現に利用されている．

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いくと最後には変数に行きつく．逆に変数からその寄与しているノードを辿るとその変数に依存している関数に行きつく．自動微分算法は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手法である．自動微分算法のうち，関数側からノードを辿る方法をtop-down算法，逆をbottom-up算法と呼び，どちらが効率的かは変数と制約式の数の大小に依存する[18]．一階微係数を求める場合，関数の数よりも変数の数が多い場合には前者が，逆の場合には後者が有利である．

計算グラフの作成は式の記述に等価である．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NLPでは特に重要となる．例えばあるプラント最適化問題（NLP）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モデリング言語の記述約10000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簡潔に記述する機能を持ち，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二階までの微係数を提供する．

NLPでは，自動微分算法の時間が実際の計算に影響する場合も多い．計算グラフが複雑な問題においては自動微分による微係数計算の手間が無視できないこと，さらに，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ことが理由として挙げられる．例えば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計算アルゴリズムの選択にも影響を及ぼす．次は信用リスク管理に必要な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]．

$$\begin{aligned} &\text{変　数：} Q\in\mathbf{R}^{18\times 18}\\ &\text{最小化：} \|Q_{year}-Q^{52}\|\\ &\text{制　約：} \sum_j Q_{ij}=1,\ Q_{ij}\geq 0 \end{aligned} \tag{10}$$

表4の最初の行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している．直線探索法は一階微係数のみを必要とするのに対して信頼領域法は二階微係数をあわせて必要とする．二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になっている．しかし，信頼領域法は二階微係数を用いているので，直線探索法に比べて解法自体の収束が速く，全体の計算時間では信頼領域法が有利となっている．表4の最後の行も類似の格付け推移確率行列の期間構造の推定問題[19]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．最初に述べたプラント最適化の問題と同様に上記の2例でも信頼領域法を適用した場合，自動微分演算に全体の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いえる．

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．上記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，計算効率を向上させることが考えられる[11]．

5. NLPの具体例

NLP（QP）を用いた定式化が有効な例としては例えば次のようなものが挙げられる．

1. マルコフ過程による信用リスクモデル
2. 債券ポートフォリオ最適化
3. プラント解析モデル
4. 化学反応系を含む生産計画モデル
5. 電力システムにおける最適潮流計算
6. 実験計画法の応用

NLPモデルの非線形性はモデルの本質的構造に由来の場合もあるが，精度的な要求によるものも多い．

表4　格付け推移確率行列推定問題の結果

			直線探索法		信頼領域法	
問題	変数	制約式	反復	計算時間	反復	計算時間
52乗根	324	18	337	2125秒	78	1616秒
期間構造	72	841	389	875秒	63	393秒

モデルの本質的な構造によるものとしては，例えば1.，2.が挙げられ，1.はマルコフ過程を記述する確率推移行列が行列積の繰り返しとなることから，2.は金利計算にlog/exp関数が現れることから非線形性が現れる．

一方，3.～6.は後者の例で，全体はよく知られたLPの構造を持つが精度を追及する過程で非線形性の導入が必要となる．3.はプラントを構成する基幹部品の状態を記述する物理量の記述から，4.は化学反応(例えば混合・燃焼）の前後における物質，例えば原油などの化石燃料の体積や性状の変化から，5.は電力に対する燃料コスト関数から，6.は仮定するフィッティングの式の次数により，それぞれ非線形性が持ち込まれる．このようなモデルは一般に大規模だが，よく知られたLPの構造に非線形関数を「はめ込んだ」形となることが多い．このような問題を解くにあたって，LP⇒NLPへの実装技術の継承は特に重要である．

NLPを実際に応用する場合に注意する点として，一般のNLPには解の一意性が保障できず，局所解が存在する場合があるため，通常の数理計画法ソフトウエアでは大域的な最適解が得られる保証がないということが挙げられる．これは多くの数理計画法ソフトウエアが，最適性に関して一次の必要条件のみしか考慮していないということに起因する．そのようなケースには実務上の要件から制約式を追加して実行可能領域を狭めて最適化を実行する，あるいは初期値を変化させて複数回の求解を繰り返すことになる．

モデルに含まれている関数の性質が既知の状態では問題の凸緩和と分枝限定法を利用することによって大域的最適解を求めることが可能な場合がある．ただし，その場合にはNLPの手続きをツール化して別の手続きから繰り返し呼ぶ手続きが必須である．その場合にはNLPの実装のソフトウエアとしてのインタフェースも重要になる．

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, 189-252, Kluwer Academic Publishers, 1996.

[6] R. C. Whaley, A. Petitet, and 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, 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.

[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，㈱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，テクニカルレポート，㈱数理システム，2002.

[12] 高橋，石原，石岡，小野，古塩，須藤，中村，「火力・揚水・水系水力の協調を考慮した需給運用計画機能の開発」，電気学会研究会資料，PE-02-130，2002.

[13] 小島政和，土谷隆，水野眞治，矢部博，「内点法」，朝倉書店，2001.

[14] 田辺隆人，「非線形最適化のアルゴリズムとソフトウエア」，最適化とアルゴリズム研究部会（SOA）発表資料，1999.

[15] A. M. Erisman and J. K. Reid, Direct Methods for sparse matrices, Oxford University Press, New York, 1986.

[16] E. Rothberg and B. Hendrickson, Sparse Matrix Ordering Methods for Interior Point Linear Programming, Manuscript, 1996 (http://www-unix.mcs.anl.gov/otc/InteriorPoint/abstracts/Rothberg-Hendrickson.html).

[17] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Ham-

marling, *Algorithm 679 : A set of Level 3 Basic Linear Algebra Subprograms,* ACM Trans. Math. Soft., 16, pp. 18-28, 1990.

[18] 伊理正夫, 久保田光一, 高速自動微分法(1), (2) 応用数理, 1, 17-35, 153-163, 1991.

[19] K. Aonuma and T. Tanabe, "An Estimation Model for Term Structure of Yield Spread", Asia-Pacific Financial Markets, 2001.

[20] A. George and J. W. H. Liu, *The evolution of the minimum degree algorithm,* SIAM Review, 31 : 1-19, 1989.

[21] Murtagh, Advanced Linear Programming, McGraw-Hill, 1981.