行列演算用言語LAMAX-S(3)

内田 智史, 本郷 茂, 八巻 直一

1. はじめに

LAMAX-S入門の最後として、数理計画法の分野から、改訂シンプレックス法と非線形方程式の求解の2つの例題を紹介します。その後、LAMAX-Sの現状と今後についてさまざまな観点からふれたいと思います。

2. 改訂シンプレックス法

改訂シンプレックス法は、その手順が行列で表現されているため、LAMAX-Sで記述するには最適な例題です。

最大化　$z = c^t x$

条　件　$Ax = b$

$x \geqq 0$

このアルゴリズムは文献[3]から引用すると次のようになります。

ステップ0

制約行列Aを、B(実行可能基底行列)、N(非基底行列)に複写する。

利益ベクトルcを、c_B, c_Nに複写する。

ステップ1

方程式 $\pi B = c_B^t$ を解いて、πを求める。

被約費用ベクトル $\overline{c}_N^t = c_N^t - \pi N$を計算する。

もし、$\overline{c}_N \leq 0$ ならステップ4へ

$\overline{c}_N$ の最大の要素の値を$\overline{c}_{j_s}$ とする。

ステップ2

方程式 $B\overline{a}_{j_s} = a_{j_s}$ を解いて$\overline{a}_{j_s}$を計算する。

もし、$\overline{a}_{j_s} \leq 0$ ならステップ5へ

方程式 $B\overline{b} = b$ を解いて$\overline{b}$を求める。

$\overline{b}_r / \overline{a}_{rj_s} = min\{\overline{b}_i / \overline{a}_{ij_s} | \overline{a}_{ij_s} > 0\}$ なるrを求める。

ステップ3

Bのr列目とNのj_s列目を入れ換える。

c_Bのr個目とC_Nのj_s個目の値を入れ換える。

ステップ1へ

ステップ4

$x_B = B^{-1}b, x_N = 0$が最適解となり終了

ステップ5

無限解となり終了

LAMAX-Sを用いてこのアルゴリズムを記述するには、少し問題があります。それはLAMAX-Sでは、大文字と小文字を文法上区別しないので、右辺ベクトル(b)と基底行列(B)をプログラム上の変数としてそのまま表現できないのです。そこで、アルゴリズムとプログラム中の各変数の対応を次のようにしました。

A	A	b	bv	c	c	B	B
N	N	c_B	cB	c_N	cN	π	pi
x_B	xB	$\overline{a}_{j_s}$	ajs	$\overline{b}$	bb	$\overline{c}_N$	cNb

LAMAX-Sによって書かれたプログラムは、ほとんどアルゴリズムそのままですので、ポイントとなるところだけを説明することにします。

このプログラムでは、要素が整数型のベクトルBindex、Nindexを宣言しています。これは、基底行列および非基底行列の変数の番号を保存するためのものです。プログラムの先頭部分で、データを各変数に代入していますが、このデータは参考文献[3]から引用しました。

それでは、このプログラムに関連したLAMAX-Sの文法事項について説明することにしましょう。LAMAX-Sでは、行列の部分を`A[i:j,k:l]`と表現し、これは行列Aのi行からj行、k列からl列までの部分行列を意味します。`i:j`や`k:l`の部分に`*`と記述するとそれはすべての範囲を意味し、`i:*`と記述するとそれは「i以降」を意味します。プログラム中で、Aの部分行列をそれぞれB、Nに代入していますが、それは次の意味です。

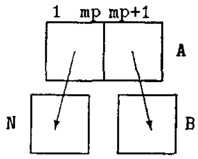

図2-1 行列Aの部分行列のコピー

`B**(-1)`という表現は、Bの逆行列を意味します。しかし、LAMAX-Sでは計算量と精度の観点から、逆行列の計算を可能な限り避ける[1]ように最適化します。`pi=cB'*B**(-1)`の計算では、実際には逆行列を求めず、行列Bを三角分解して連立一次方程式の解として求めます。また、`solve`文を用いれば、明示的に連立一時方程式を解くことができます。

うちだ　さとし　神奈川大学
〒221 横浜市神奈川区六角橋3—27—1
ほんごう　しげる　専修大学
やまき　なおかず　㈱システム計画研究所


```
solve A * x = b
```

で連立一次方程式 $Ax=b$ を解いて x を求めます。

アルゴリズム中の $\boxed{\text{ステップ 2}}$ で、

$$\bar{b}_r/\bar{a}_{rj_s} = min\{\bar{b}_i/\bar{a}_{ij_s}|\bar{a}_{ij_s} > 0\}$$

なる r を求める部分がありますが、これをLAMAX-Sでは次のようにallmin関数を用いて処理します。allmin関数は行列の最小の値を返す関数です。allmin(A) とすると、行列 A の最小の要素の値を返しますが、これにいろいろな条件を付加することができます。たとえば、allmin(A, +diagonal) では、行列 A の対角要素の中から最小の要素を探します。それでは、次の命令は何でしょうか。ここで、A と B は同じ寸法です。

```
r = allmin( A, +pos, B .gt. 0.0 )
```

+pos の場合、第 3 引数に記述された論理演算が行列の要素単位ごとに演算され、その結果が真となる要素の位置と同じ位置の行列 A の要素が検索対象となります。つまり、この例では、A の最小要素を検索しますが、検索対象となるのは、B の要素が 0 よりも大きい位置だけとなります。したがって、r を求めるには、

```
r = allmin( bb%ajs, +pos, ajs.gt.0.0 )
```

とすればよいのです。ここで%は、要素ごとの除算を意味します。

<=>は、行列やベクトル、要素を交換する命令です。A<=>B で A と B が交換されます。

このプログラムでは、変数の番号と値を並べて表示するために、| Bindex, xB | という行列を合成しています。これによって 1 列目にはベクトル Bindex、2 列目には xB からなる行列が合成されます。

3. ニュートン法

ORモデルには、非線形方程式に帰着するものがしばしば現れます。そのとき、求める解を得るための有効な手法として、ニュートン法があります。ここでは、非線形方程式 $f(x)=0$ の解を求めるプログラムを紹介します。ただし、f, x は n 次元ベクトルとします。また、以下の説明において、$g(x)$ は f の x におけるヤコビ行列とします。このアルゴリズムは次のようになります。

$\boxed{\text{ステップ 0}}$

$k=0$ とし、初期点を x_0 とする。

$\boxed{\text{ステップ 1}}$

$f(x_k)$ が十分 0 に近ければステップ 5 へ。

$\boxed{\text{ステップ 2}}$

連立一次方程式 $g(x_k)d=-f(x_k)$ を解いて d を求める。

$\boxed{\text{ステップ 3}}$

$x_{k+1}=x_k+d$ を計算し近似解を更新する。

$\boxed{\text{ステップ 4}}$

$k=k+1$ とし、ステップ 1 へ

$\boxed{\text{ステップ 5}}$

解 x_k を表示して終了

プログラムは、次のようになります。ところで、$f(x)$ はベクトル関数であり、$g(x)$ は行列関数です。たとえば、このプログラムでは、$f(x)=(x_1^2+x_2^2-25, x_1+x_2-7)^t$ としています。

行列 (ベクトル) を返す関数では、構造の指定を function 文に記述しますが、その寸法には必ず*を記述します。

この例では、ニュートン法の数学的表現の言語上での再現性がきわめて良いことが示されています。行列表現の導入と、行列関数の概念の導入がアルゴリズムの表現には欠かせない要素であることが認識される例といえましょう。

4. LAMAX-S の今後と関連研究

この連載で、6 つのプログラム (ヤコビ法 (固有値・固有ベクトル)、重回帰分析、主成分分析、改訂シンプレックス法、ニュートン法 (非線形方程式)) の例を紹介してきました。かなり難しい内容の例題でも、LAMAX-S で記述すればひじょうに簡潔に記述できることがご理解いただけたのではないかと思います。今までにかなりの例題を LAMAX-S で記述してきましたが、ほとんど 1 枚の A4 用紙にプログラムが納まりました。また、プログラミングに要する時間ですが、他の言語で行うのに比べて 5 倍から 10 倍程度のスピードでプログラミングを行うことができました。また、何よりもアルゴリズムそのままの形でプログラム上に記述できるため、アルゴリズムのイメージが強烈に浮び上がり、教育的効果が高いことがわかりました。実際、LAMAX-S を用いて学部学生にプログラムを組ませたところかなり好評でした。

このように教育的な立場から、LAMAX-S の利用は大いに効果があると考えられますが、我々は実務家や OR

研究者の実際の研究活動にも LAMAX-S は十分に活用できると考えています。現在、実際に我々が提供できる LAMAX-S 処理系は、パソコン版だけであり、多くの文法上の制限があります。近いうちに、我々はこの制限を極力除いてさらにワークステーションに移植する予定です。

このような言語の設計・開発では利用者の意見が大きなウエイトを占めますので、このワークステーション版処理系を用いて、実際に多くの方々に利用していただき、LAMAX-S の諸機能をさらに高めていこうと考えています。現在の LAMAX-S の文法検討事項として、(1) 集合の導入、(2) インデックス行列の導入、(3) 関数の微分機能の導入、などがあります。

一方、スーパコンピュータに対応可能な本格的な LAMAX-S 処理系もその開発が進められています。この新処理系では、多様なライブラリを利用でき、さらに自動チューニング機能および数学的特性や数学的記述をベースにして、計算量や記憶量を減らす処理 (これを私達は数学的最適化機能と呼んでいます) も含まれています。この新処理系は、現在の PC98 版よりも生成される FORTRAN コードの質を大幅に向上させていて、スーパコンピュータだけでなくワークステーション・パソコンにも利用可能になります。

現在、この新処理系の開発と平行して、自動チューニング機能を実現するために、各種コンピュータに対するチューニング特性測定システムを開発しています。これは、LAMAX-S が各コンピュータの特性に合わせた (いわゆるチューニングされた)FORTRAN コードを生成できるようにするために、さまざまなサンプルテストプログラムを生成し、実際にコンピュータにかけ、チューニングの特性を調べるものです。この測定結果を元に LAMAX-S のチューニング知識ベースを構築し、自動チューニングを実現する予定です。

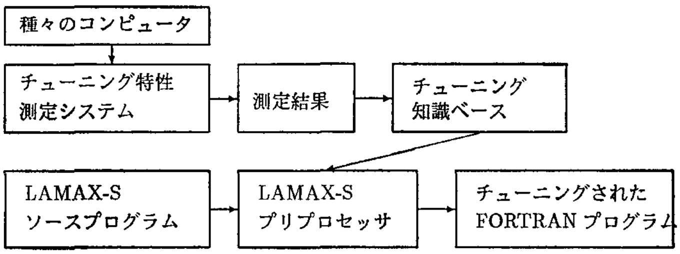

図 4-1 チューニング知識ベースの利用

このような技術を結集し、私たちは、実用的な LAMAX-S 処理系を目指してゆこうと考えています。

さらに、LAMAX-S の「その後」のプログラミング環境についても研究が進められています。LaTeX で書かれたアルゴリズムを LAMAX-S のプログラムに変換する処理系や LAMAX-S のソースコードを生成する数学ソフトウェアプログラミング支援のための CASE ツールなどが、プロトタイプではありますが完成しています。次のアルゴリズムは LaTeX によって書かれたヤコビ法による連立一次方程式の解法ですが、特殊なトランスレータによって、この LaTeX のファイルを LAMAX-S プログラムに変換し実行させることができます。

近い将来、現在あるようなアスキー文字をベースにしたテキスト中心のプログラミング言語は姿を消すかも知れません。私たちの将来の研究は、プログラミング言語そのものではなく、数学的ソフトウェアのプログラミング環境やその実現技術に関するものになって行くと考えています。

ヤコビ法 (LaTeX → LAMAX-S)

$\boxed{\text{変数宣言}}$

n:整定数=5
ϵ:実定数=10^{-6}
A:行列 $[n \times n]$
B:ベクトル $[n]$
D:対角行列 $[n \times n]$
L:狭義下三角行列 $[n \times n]$
U:狭義上三角行列 $[n \times n]$
$X, NewX$:ベクトル $[n]$

$\boxed{\text{Step. 1}}$ A, Bを入力する。

$\boxed{\text{Step. 2}}$ AをL, D, Uに分割する。

$\boxed{\text{Step. 3}}$ 初期点の設定

$$X = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Step. 4 方程式を解く

方程式 $D\ NewX = B + (L+U)\ X$ を $NewX$ について解く。

$diff = (NewX - X)$ の絶対値が最大の行列要素の値

もし $diff \leq \epsilon$ ならば、Step. 6 へ行く。

もし $mc > maxcnt$ ならば、Step. 5 へ行く。

$X = NewX$

$mc = mc + 1$

Step. 4 へ行く。

Step. 5 「収束せず」と表示し、プロダグラムを終了する。

Step. 6 $NewX$ を表示し、プログラムを終了する。

なお、PC98 版 LAMAX-S 処理系入手希望の方はシステム計画研究所 (03-5489-0211) にお問い合わせください (詳細は本連載 (1),Vol.38, No.2, pp.88-91 をご覧下さい)。

参考文献

1. 二宮市三：科学技術計算への二つの提案、応用数理、Vol.2, No.1, 1992
2. 内田智史、八巻直一、本郷茂、井上美明、唐澤豊：行列の数学的特性を重視した言語 LAMAX-S の設計思想とその処理系について、日本応用数理学会論文誌、 Vol. 2, No. 3, pp.155-168, 1992
3. 今野浩：線形計画法、日科技連、pp.60-63


```
c       改訂シンプレックス法
        parameter(mp=3,np=6,lp=3)
        real:matrix[mp,np] A      <* 制約行列
        real:vector[mp]    bv     <* 右辺ベクトル
        real:vector[np]    c      <* 利益ベクトル
        real:matrix[mp,mp] B      <* 基底行列
        real:matrix[mp,lp] N      <* 非基底行列
        real:vector[mp]    cB     <*
        real:vector[lp]    cN     <*
        real:rvector[mp]   pi     <* 単体乗数ベクトル
        real:vector[mp]    xb     <* 最適解
        real:vector[mp]    ajs
        real:vector[mp]    bb
        real:vector[lp]    cNb    <* 被約費用ベクトル
        integer:vector[mp] mask
        integer:vector[mp] Bindex    <* 基底行列のイン
                                        デックス
        integer:vector[lp] Nindex    <* 非基底行列のイン
        integer js,r                    デックス
c
c       データの設定
        A = | 1, 1, 2, 1, 0, 0 !
     -      ! 2, 0, 2, 0, 1, 0 !
     -      ! 2, 1, 3, 0, 0, 1 |
        bv = | 4, 5, 7 |'
        c  = | 3, 2, 4, 0, 0, 0 |'
        Bindex = | 4, 5, 6 |'
        Nindex = | 1, 2, 3 |'
c
c ステップ 0  各種データの初期化
        B = A[*,mp+1:*]
        N = A[*,1:mp]
        cB = c[mp+1:*]
        cN = c[1:mp]
c ステップ 1
10000 continue
        pi  = cB'*B**(-1)
        cNb = ( cN' - pi*N )'
        if(allmax(cNb).le.0.0) goto 99999
        js = almaxr(cNb)
c ステップ 2
        solve B * ajs = a[*,js]
        if(allmax(ajs).le.0.0) goto 99998
        solve B * bb = bv
        r = allmin( bb%ajs , +pos, ajs.gt.0.0)
c ステップ 3  基底の入れ替え
        B[*,r]      <=> N[*,js]
        cB[r]       <=> cN[js]
        Bindex[r] <=> Nindex[js]
        go to 10000
c ステップ 4
99999 continue
        write(*,*) '終了'
        solve B * xb = bv
        call mprint(| Bindex, xb |)
        stop
c ステップ 5
99998 continue
        write(*,*) '解：無限解'
        end
```

```
c       ニュートン法
        parameter(n=2, eps=1.e-3)
        real:vector[n] x
        real:vector[n] d
        real:vector[*] f
        real:matrix[*,*] G
c
        k = 0
        x = | 5, 8 |'
c
   10 continue
           if( absmax( f(x) ) .lt. eps ) go to 20
           solve G(x) * d = -f(x)
           x = x + d
           k = k + 1
        go to 10
c
   20 continue
        call mprint(x)
        stop
        end
c-----------------------------------------------
        real:vector[*] function f(x)
        real:vector[2] x
c
        f = | x' * x - 25,  x[1]+x[2]-7 |'
        return
        end
c-----------------------------------------------
        real:matrix[*,*] function G(x)
        real:vector[2] x
c
        G = | 2*x[1],  2*x[2] !
     -      !      1,       1 |
        return
        end
```