社会システム分析のための統合化プログラム３

― 線形計画法・待ち行列シミュレーション ―

増川純一，福井正康，田口賢士

福山平成大学経営学部経営情報学科

概要

我々は主として教育を目的に、社会システム分析に用いられる様々な手法を統一的に扱うプログラムを作成している。今回は、線形計画法と待ち行列シミュレーションについて、用いられた理論とプログラムの利用法を詳述する。線形計画法ではシンプレックス法に関する諸概念を、待ち行列では理論とシミュレーション手法をあわせて学習することを目的とする。

キーワード

線形計画法，シンプレックス法，待ち行列理論，シミュレーション，ソフトウェア，統合化プログラム

1章　はじめに

これまで教育を１つの目的として、統計学、OR、その他の分析を統合的に扱うプログラムの枠組みを作成し[1)]、産業連関分析、KSIM、AHP についてMS-Windows 上の Visual Basic によるプログラムを紹介してきたが[2)]、この論文では線形計画法と待ち行列シミュレーション[3)-5)]について、用いられた理論とプログラムの利用法を解説する。

線形計画法に関してはこれまでさまざまなソフトウェアが作られ、現実問題としても数千を超える変数と制約式を持つ問題が解かれている。このような状況において、線形計画法のプログラムを新たに作る意味は、社会システム分析用の統合アプリケーションとしては避けて通れないという消極的な理由の他に、線形計画法の理論を具体的な事例で如何に効果的に学ぶかという教育的な問題に解答を試みることにある。また、現実問題として、限られた予算の教育現場において高価で高機能な線形計画パッケージをごく簡単な問題だけに適用することは、経済的にも負担が大きいし、機能が多い分だけ初心者にとって焦点が不鮮明に成りがちである。それゆえフリーソフトとして、機能を限定し、初心者が利用しやすいように、選択操作だけで処理が実行出来るようなプログラムを作ることは、あながち無駄なこととは思えない。今回は線形計画法の中でもシンプレックス法に焦点を絞り、出来るだけ簡単にその構造と手法が理解出来るようにプログラムを作成した。

待ち行列に関する通常の教育プログラムでは、理想的な条件を仮定して、理論的に平衡状態や過渡状態を求めることが主題となる事が多い。一方、現実の問題では、条件としての単位時間当りの到着人数の分布やサービス時間の分布は必ずしも理想的な分布とはならず、状況に合わせてシミュレーションモデルを作らなければならない。それゆえ理論とあわせてシミュレーション手法を習得しておくことは応用上重要と考えられる。また、待ち行列の時間変動の様子が視覚化されれば、理論に関する理解を深める上でも有効である．そこで我々は、理論とシミュレーションモデルとの橋渡しをする教育用ツールの作成を考えた。このツールでは、到着人数分布やサービス時間分布、待ち行列の長さ、窓口の数、実行時間等の計算条件を設定し、計算を実行することにより、それぞれの条件に応じた待ち行列の変動をあたかも現場で観測しているように追跡することが出来る．また、シミュレーションの結果を理論値と比較することもできる。我々はこのツールによってサービスシステムの設計に柔軟に対応できる人材が育成されることを期待している．

２章では線形計画法を３章では待ち行列を取り上げ、４章で問題点や今後の課題等について議論する。プログラムは線形計画法を福井が、待ち行列を増川が担当した。

2章　線形計画法

2.1　基礎理論

線形計画法は、一般に (2.2), (2.3) 式の制約条件のもとで、(2.1) 式の目的関数を最大化（最小化）する問題を解決するための手法である。

目的関数　　$z={}^t\mathbf{c}^o\mathbf{x}^o$　最大化（最小化）,　　(2.1)

制約条件　　$\mathbf{A}_1\mathbf{x}^o \le \mathbf{b}_1$,　　(2.2a)

$\mathbf{A}_2\mathbf{x}^o \ge \mathbf{b}_2$,　　(2.2b)

$\mathbf{A}_3\mathbf{x}^o = \mathbf{b}_3$,　　(2.2c)

$\mathbf{x}^o \ge \mathbf{0}$.　　(2.3)

ここに、$\mathbf{x}^o(n_0\times 1)$ は（2.3)式によって各要素が非負に制限された変数行列であり、$\mathbf{c}^o(n_0\times 1)$, $\mathbf{b}_i(m_i\times 1)\ge\mathbf{0}$, $\mathbf{A}_i(m_i\times n)$ は係数行列である。

制約条件が (2.2a) だけの場合は、スラック変数を加え、容易に初期実行可能基底解が与えられるので、すぐにピボット操作による計算が実行出来るが、(2.2b) または (2.2c) を含む場合は、一般に実行可能基底解を得るために、2段階法を適用する。即ち、スラック変数 $\mathbf{x}_1^s(m_1\times 1)\ge\mathbf{0}$, $\mathbf{x}_2^s(m_2\times 1)\ge\mathbf{0}$ と人為変数 $\mathbf{x}_2^a(m_2\times 1)\ge\mathbf{0}$, $\mathbf{x}_3^a(m_3\times 1)\ge\mathbf{0}$ を加え、(2.2)式を変形する。

$\mathbf{A}_1\mathbf{x}^o + \mathbf{x}_1^s = \mathbf{b}_1$,　　(2.4a)

$\mathbf{A}_2\mathbf{x}^o - \mathbf{x}_2^s + \mathbf{x}_2^a = \mathbf{b}_2$,　　(2.4b)

$\mathbf{A}_3\mathbf{x}^o + \mathbf{x}_3^a = \mathbf{b}_3$.　　(2.4c)

第1段階の目的関数は、

$w={}^t\mathbf{e}_2\mathbf{x}_2^a+{}^t\mathbf{e}_3\mathbf{x}_3^a$　最小化 ,　　(2.5)

のように書ける。ここに、$\mathbf{e}_i(m_i\times 1)$ は、全ての成分が 1 の行列である。

(2.5)式は基底形式でないので、(2.4b), (2.4c)式を加え、基底形式に変形する。

第1段階目的関数　　$w+{}^t\mathbf{d}\mathbf{x}={}^t\mathbf{e}\mathbf{b}$,　　w：最小化　　(2.6)

第2段階目的関数　　$z-{}^t\mathbf{c}\mathbf{x}=0$,　　z：最大化（最小化）　　(2.7)

制約条件　　$\mathbf{A}\mathbf{x}=\mathbf{b}$,　$\mathbf{x}\ge\mathbf{0}$.　　(2.8)

ここに、記号については以下にまとめる。

$$\mathbf{x}(n\times 1)=\begin{pmatrix}\mathbf{x}^o\\ \mathbf{x}_1^s\\ \mathbf{x}_2^s\\ \mathbf{x}_2^a\\ \mathbf{x}_3^a\end{pmatrix},\quad \mathbf{c}(n\times 1)=\begin{pmatrix}\mathbf{c}^o\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{pmatrix},\quad \mathbf{d}(n\times 1)=\begin{pmatrix}{}^t\mathbf{A}_2\mathbf{e}_2+{}^t\mathbf{A}_3\mathbf{e}_3\\ \mathbf{0}\\ -\mathbf{e}_2\\ \mathbf{0}\\ \mathbf{0}\end{pmatrix},$$

$$\mathbf{A}(m\times n)=\begin{pmatrix}\mathbf{A}_1 & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{A}_2 & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0}\\ \mathbf{A}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{pmatrix},\quad \mathbf{b}(m\times 1)=\begin{pmatrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \mathbf{b}_3\end{pmatrix},\quad \mathbf{e}(m\times 1)=\begin{pmatrix}\mathbf{0}\\ \mathbf{e}_2\\ \mathbf{e}_3\end{pmatrix},$$

$$n=n_0+m_1+2m_2+m_3,\quad m=m_1+m_2+m_3. \tag{2.9}$$

ピボット操作を繰り返し、第 1 段階で最適解が存在すれば、$\mathbf{x}_2^a=\mathbf{0}$，$\mathbf{x}_3^a=\mathbf{0}$，$w=0$ の解を得る。このとき、その最適解についての基底行列を$\mathbf{B}_1$、それに対応する係数$\mathbf{c}$と$\mathbf{d}$の基底部分を$\mathbf{c}_{B_1}$，$\mathbf{d}_{B_1}$とすると、(2.6)~(2.8)式は以下となる。

第 1 段階目的関数　$$w+({}^t\mathbf{d}-{}^t\mathbf{d}_{B_1}\mathbf{B}_1^{-1}\mathbf{A})\mathbf{x}={}^t\mathbf{e}\mathbf{b}-{}^t\mathbf{d}_{B_1}\mathbf{B}_1^{-1}\mathbf{b}\ (=0), \tag{2.10}$$

第 2 段階目的関数　$$z-({}^t\mathbf{c}-{}^t\mathbf{c}_{B_1}\mathbf{B}_1^{-1}\mathbf{A})\mathbf{x}={}^t\mathbf{c}_{B_1}\mathbf{B}_1^{-1}\mathbf{b}, \tag{2.11}$$

制約条件　$$\mathbf{B}_1^{-1}\mathbf{A}\mathbf{x}=\mathbf{B}_1^{-1}\mathbf{b},\quad \mathbf{x}\ge\mathbf{0}. \tag{2.12}$$

第 1 段階の最適解は第 2 段階の初期可能基底解になっている。そこで、人為変数を消去して次元を縮小し、新たに、以下の問題を考える。

目的関数　$$z-{}^t\mathbf{c}^{(1)}\mathbf{x}={}^t\mathbf{c}_{B_1}\mathbf{B}_1^{-1}\mathbf{b},\qquad z：最大化（最小化） \tag{2.13}$$

制約条件　$$\mathbf{A}^{(1)}\mathbf{x}=\mathbf{b}^{(1)},\quad \mathbf{x}\ge\mathbf{0}. \tag{2.14}$$

記号については改めて以下にまとめる。

$$\mathbf{A}^{(1)}=\mathbf{B}_1^{-1}\mathbf{A},\quad \mathbf{b}^{(1)}=\mathbf{B}_1^{-1}\mathbf{b},\quad {}^t\mathbf{c}^{(1)}={}^t\mathbf{c}-{}^t\mathbf{c}_{B_1}\mathbf{B}_1^{-1}\mathbf{A},$$

$$\mathbf{A}(m\times n)=\begin{pmatrix}\mathbf{A}_1 & \mathbf{I} & \mathbf{0}\\ \mathbf{A}_2 & \mathbf{0} & -\mathbf{I}\\ \mathbf{A}_3 & \mathbf{0} & \mathbf{0}\end{pmatrix},\quad \mathbf{x}(n\times 1)=\begin{pmatrix}\mathbf{x}^o\\ \mathbf{x}_1^s\\ \mathbf{x}_2^s\end{pmatrix},\quad \mathbf{c}(n\times 1)=\begin{pmatrix}\mathbf{c}^o\\ \mathbf{0}\\ \mathbf{0}\end{pmatrix},$$

$$n=n_0+m_1+m_2,\quad m=m_1+m_2+m_3. \tag{2.15}$$

(2.13), (2.14)の問題に対して、最適解が存在する場合には、ピボット操作を行った後、最終的に以下の式を得る。

目的関数　$$z-{}^t\mathbf{c}^{(2)}\mathbf{x}={}^t\mathbf{c}_{B_1}\mathbf{b}^{(1)}+{}^t\mathbf{c}_{B_2}^{(1)}\mathbf{b}^{(2)}, \tag{2.16}$$

制約条件　$$\mathbf{A}^{(2)}\mathbf{x}=\mathbf{b}^{(2)}, \tag{2.17}$$

$$\mathbf{A}^{(2)}=\mathbf{B}_2^{-1}\mathbf{A}^{(1)},\quad \mathbf{b}^{(2)}=\mathbf{B}_2^{-1}\mathbf{b}^{(1)},\quad {}^t\mathbf{c}^{(2)}={}^t\mathbf{c}^{(1)}-{}^t\mathbf{c}_{B_2}^{(1)}\mathbf{B}_2^{-1}\mathbf{A}^{(1)}, \tag{2.18}$$

ここに、$\mathbf{B}_2$ は $\mathbf{A}^{(1)}$ についての基底行列である。

これらの式は、最終的な基底変数に対応する基底行列$\mathbf{B}\ (=\mathbf{B}_1\mathbf{B}_2)$を用いると、以下のように書けることにも注意しておく必要がある。

目的関数　$$z-{}^t\hat{\mathbf{c}}\mathbf{x}={}^t\mathbf{c}_B\mathbf{B}^{-1}\mathbf{b}, \tag{2.19}$$

制約条件　$$\hat{\mathbf{A}}\mathbf{x}=\mathbf{B}^{-1}\mathbf{b}, \tag{2.20}$$

$${}^t\hat{\mathbf{c}}\equiv{}^t\mathbf{c}^{(2)}={}^t\mathbf{c}-{}^t\mathbf{c}_B\mathbf{B}^{-1}\mathbf{A},\quad \hat{\mathbf{A}}\equiv\mathbf{A}^{(2)}=\mathbf{B}^{-1}\mathbf{A}. \tag{2.21}$$

これより基底変数 $\mathbf{x}_B$ と非基底変数 $\mathbf{x}_N$ に対して、最適解は、$\hat{\mathbf{x}}_B=\mathbf{B}^{-1}\mathbf{b}$，$\hat{\mathbf{x}}_N=\mathbf{0}$　となる。

その際、目的関数は $\hat{z}={}^t\mathbf{c}_B\mathbf{B}^{-1}\mathbf{b}$ となる。

線形計画問題 (2.1)~(2.3) に対して、一般性を失わず、制約式右辺の非負性 $\mathbf{b}\geq\mathbf{0}$ を除き、また変数の非負性 $\mathbf{x}\geq\mathbf{0}$ も一部取り除いて、

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_{11}(m_1\times n_1) & \mathbf{A}_{12}(m_1\times n_2)\\ \mathbf{A}_{21}(m_2\times n_1) & \mathbf{A}_{22}(m_2\times n_2)\end{pmatrix},\ \mathbf{b}=\begin{pmatrix}\mathbf{b}_1(m_1\times 1)\\ \mathbf{b}_2(m_2\times 1)\end{pmatrix},\ \mathbf{c}=\begin{pmatrix}\mathbf{c}_1(n_1\times 1)\\ \mathbf{c}_2(n_2\times 1)\end{pmatrix},$$
$$\mathbf{x}=\begin{pmatrix}\mathbf{x}_1(n_1\times 1)\\ \mathbf{x}_2(n_2\times 1)\end{pmatrix},\ \mathbf{y}=\begin{pmatrix}\mathbf{y}_1(m_1\times 1)\\ \mathbf{y}_2(m_2\times 1)\end{pmatrix}, \qquad (2.22)$$

の記号を用いると、主問題と双対問題はお互いに以下のように表現することが出来る。

主問題（双対問題）		双対問題（主問題）
$z={}^t\mathbf{cx}$　最大化,		$z'={}^t\mathbf{by}$　最小化,
$\mathbf{A}_{11}\mathbf{x}_1+\mathbf{A}_{12}\mathbf{x}_2\leq\mathbf{b}_1$,	$\Leftrightarrow$	${}^t\mathbf{A}_{11}\mathbf{y}_1+{}^t\mathbf{A}_{21}\mathbf{y}_2\geq\mathbf{c}_1$,
$\mathbf{A}_{21}\mathbf{x}_1+\mathbf{A}_{22}\mathbf{x}_1=\mathbf{b}_2$,		${}^t\mathbf{A}_{12}\mathbf{y}_1+{}^t\mathbf{A}_{22}\mathbf{y}_2=\mathbf{c}_2$,
$\mathbf{x}_1\geq\mathbf{0}$.		$\mathbf{y}_1\geq\mathbf{0}$.

(2.23)

これらの問題の間には、最適解 ⇔ 最適解，無限解 ⇒ 不能，不能 ⇒ 無限解または不能，の関係が存在するが、最適解の場合、双方の目的関数の値は等しくなる。

$$\hat{z}={}^t\mathbf{c}_B\hat{\mathbf{x}}_B={}^t\mathbf{c}_B\mathbf{B}^{-1}\mathbf{b}={}^t\hat{\mathbf{y}}\,\mathbf{b}=\hat{z}'. \qquad (2.24)$$

ここに、変数 $\mathbf{y}$ の最適解 $\hat{\mathbf{y}}={}^t\mathbf{B}^{-1}\mathbf{c}_B$ は双対価格と呼ばれる。

次に、パラメータの変化に対する最適解の安定性を吟味する感度分析について考える。目的関数の係数の微小変化によって、最適解の係数は ${}^t\hat{\mathbf{c}}+{}^t\Delta\hat{\mathbf{c}}={}^t\hat{\mathbf{c}}+{}^t\Delta\mathbf{c}-{}^t\Delta\mathbf{c}_B\mathbf{B}^{-1}\mathbf{A}$ のように変化する。最大化の場合これが非正（最小化では非負）のままだと基底変数の組が維持される。即ち、非基底変数の係数について ${}^t\hat{\mathbf{c}}_R+{}^t\Delta\mathbf{c}_R-{}^t\Delta\mathbf{c}_B\mathbf{B}^{-1}\mathbf{R}\leq\mathbf{0}$（最小化では非負）を満たしていなければならない。ここに、$\mathbf{R}$ は行列 $\mathbf{A}$ の非基底部分とする。この関係より、係数変化の範囲として以下の条件を得る。但し、最小化の場合は不等号の向きが逆になることに注意する。

$$\Delta\mathbf{c}_R\leq-\hat{\mathbf{c}}_R,$$
$$(\Delta\mathbf{c}_B)_i\geq(\hat{\mathbf{c}}_R)_j\Big/(\mathbf{B}^{-1}\mathbf{R})_{ij},\quad\text{for}\ (\mathbf{B}^{-1}\mathbf{R})_{ij}>0,$$
$$(\Delta\mathbf{c}_B)_i\leq(\hat{\mathbf{c}}_R)_j\Big/(\mathbf{B}^{-1}\mathbf{R})_{ij},\quad\text{for}\ (\mathbf{B}^{-1}\mathbf{R})_{ij}<0,$$
$$\text{制限なし}\qquad\text{for}\ (\mathbf{B}^{-1}\mathbf{R})_{ij}=0. \qquad (2.26)$$

次に、係数 $\mathbf{b}$ の変化に対して、基底変数の組が変化しない範囲は基底変数の最適解が非負を維持する範囲 $\hat{\mathbf{x}}_B+\Delta\hat{\mathbf{x}}_B=\hat{\mathbf{x}}_B+\mathbf{B}^{-1}\Delta\mathbf{b}\geq\mathbf{0}$ である。これより以下の条件を得る。

$$(\Delta\mathbf{b})_i\geq-(\hat{\mathbf{x}}_B)_j\Big/(\mathbf{B}^{-1})_{ji},\quad\text{for}\ (\mathbf{B}^{-1})_{ij}>0,$$
$$(\Delta\mathbf{b})_i\leq-(\hat{\mathbf{x}}_B)_j\Big/(\mathbf{B}^{-1})_{ji},\quad\text{for}\ (\mathbf{B}^{-1})_{ij}<0,$$
$$\text{制限なし}\qquad\text{for}\ (\mathbf{B}^{-1})_{ij}=0. \qquad (2.27)$$

2.2　プログラムの動作

プログラムの操作法は極めて簡単である。エディター内のデータ形式は図 2.1 に示す。表の最上部（0 行目）は変数名、最左部（0 列目）は行番号であるが、これがなくても特に問題はない。1 行目は目的関数の係数である。最大化と最小化の区別は 1 行目最右部に、MAX または MIN の記号（大文字・小文字は区別しない）を入力して決定する。2 行目以降は制約式である。不等号・等号については BASIC で利用されるものを用いるが、不等号に等号が付いたものと付かないものは区別しない。分析の欄には線形計画法、備考の欄には最小化を表す記号と変数の数及び制約式の数を表すものが入力されているが、これらは覚書であって何も書かなくても差し支えない。

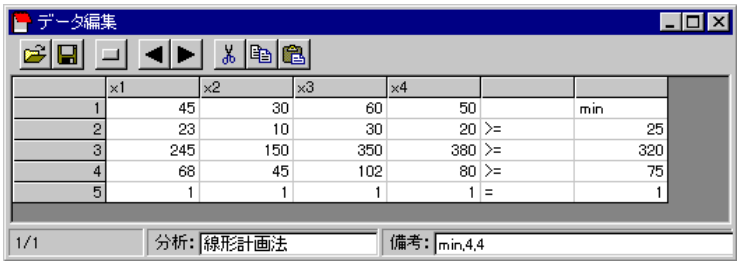

図 2.1 線形計画法のデータ形式

図 2.2 線形計画法のメニュー

親フォームの分析メニューの中から、線形計画法を選択すると、図 2.2 のようなメニューが表示される。その際、変数と制約式の部分は自動的に記入される。これは、エディターの行数と列数から求められるので、データに空白行（列）があってはならない。

メニューの「simplex 表」は、図 2.1 の線形計画問題から、標準的なシンプレックス表を作成するためのもので、以下の規則に従う。

1）最小化問題は目的関数に –1 を掛けて最大化問題に直す。
2）右辺が負の場合、両辺に -1 を掛けて不等号の向きは反対にする。
3）変数に非負の条件がない場合、変数を「変数名_1」と「変数名_2」の２つの非負変数に分け、元の変数をこれらの引き算で表す。
4）制約式の不等号が「<=」の場合、スラック変数を加える。
5）制約式の不等号が「>=」の場合、スラック変数を引き、人為変数を加える。
6）制約式が等号「=」の場合、人為変数を加える。
7）人為変数を加えた場合は、２段解法の目的関数を加える。

後に例が示されるが、変数が非負かどうかを表すために、非負条件が付かない場合、代表的な線形計画法パッケージソフトウェア LINDO[6] に習って変数名の最後に「!」記号を加えること

初期シンプレックス表

	x1	x2	x3	x4	SL1	SL2	SL3	AR1	AR2	AR3	AR4		
<-W>	-337	-206	-483	-481	1	1	1	0	0	0	0	=	-421
<-Z>	45	30	60	50	0	0	0	0	0	0	0	=	0
AR1	23	10	30	20	-1	0	0	1	0	0	0	=	25
AR2	245	150	350	380	0	-1	0	0	1	0	0	=	320
AR3	68	45	102	80	0	0	-1	0	0	1	0	=	75
AR4	1	1	1	1	0	0	0	0	0	0	1	=	1

図 2.3 初期シンプレックス表

にする。実行結果は、図 2.3 に示す。

ここにスラック変数には「SL 番号」、人為変数には「AR 番号」のように、自動的に変数名が付けられる。目的関数は Z で表し、２段階法の場合は、第１段階の目的関数として W を導入する。ここでは、第１段階も第２段階も最小化問題であるので、最大化問題にするために W と Z 両方に –1 が掛かっている。最小化問題はそのままでも計算上特に問題はないが、ここでは統一性を重視した。

図 2.2 のメニュー画面で「Pivot 操作の順次出力」を選択すると、シンプレックス法の計算過程を１ステップ毎表示する。ピボット操作のステップは、初期を 0 として、２段階法の場合も継続して数えている。続けて選択することによって、利用者は基底変数の移り変わり等を簡単に見ることが出来る。最終的なシンプレックス表が表示された後は最初の段階に戻る。

最終シンプレックス表 STEP 4

	x1	x2	x3	x4	SL1	SL2	SL3		
<-Z>	0	6.5079	0	0	1.1905	0.0635	0	=	-52.1429
SL3	0	-25.381	0	0	-2.6429	-0.1476	1	=	9.3571
x4	0	0.5556	0	1	0.0833	-0.0056	0	=	0.25
x3	0	-1.619	1	0	-0.1071	-0.0024	0	=	0.3929
x1	1	2.0635	0	0	0.0238	0.0079	0	=	0.3571

図 2.4 最終結果のシンプレックス表

最終結果

```
線形計画法 最適解  STEP 4

目的関数最小値  52.1429

  変数          最適解        被約費用
    x1          0.3571          0.0000
    x2          0.0000          6.5079
    x3          0.3929          0.0000
    x4          0.2500          0.0000

制約式          余剰          双対価格
     1          0.0000         -1.1905
     2          0.0000         -0.0635
     3          9.3571          0.0000
     4          0.0000         -2.0635

係数範囲
  変数          現在値        係数上限        係数下限
    x1         45.0000          3.1538        -infinity
    x2         30.0000        infinity         -6.5079
    x3         60.0000        infinity         -4.0196
    x4         50.0000         11.7143        -11.4286

右辺範囲
制約式          現在値        係数上限        係数下限
     1         25.0000          3.0000         -3.5405
     2        320.0000         45.0000        -45.0000
     3         75.0000          9.3571        -infinity
     4          1.0000          0.1289         -0.1023
```

図 2.5 最終結果の出力

計算の途中で、解の存在しない場合や、無限大の解が存在する場合はメッセージにより知らせる。その際、利用者はシンプレックス表により、その理由を見ることが出来る。

メニューで「結果出力」を選択すると、最終的なシンプレックス表と実行結果が図 2.4、図 2.5 のように表示される。

最終結果の出力は、LINDO の出力に習った。以下にその定義を与える。

「最適解」は変数 $\mathbf{x}^o$ の最適値であり、「被約費用」は目的関数における $\mathbf{x}^o$ の係

数である。「余剰」は制約式の左辺と右辺との差を表している。これは、初期可能基底変数の値であるため、スラック変数の最適値と等号制約の人為変数の値（= 0）である。「双対価格」は ${}^t\pi = {}^t\mathbf{c}_B\mathbf{B}^{-1}$ で定義され、双対問題の最適解でもあるが、制約式の右辺値の微小変化における目的関数の変化率としても解釈される。

感度分析において、基底変数の組が維持される目的関数の係数の範囲は、(2.26)式を用いて図 2.5 の「係数範囲」の部分に、また右辺係数の範囲は、(2.27)式を用いて「右辺範囲」の部分に表示される。

図 2.6a 主問題

図 2.6b 双対問題

主問題に対する双対問題は、図 2.2 のメニュー画面で「双対問題生成」を選択すると、エディター上に２ページ目として生成される。双対問題は同じメニュー上のオプションボタンを選択することにより、主問題と同じように解くことが出来る。これによって実際の数値で、主問題と双対問題の関係を確認することが出来る。

変数名 x2!, y2! のように後に感嘆符が付いた変数があるが、これは以前に説明したように、非負条件の付かない変数である。等号条件に対応する双対問題の変数がこれに相当する。

3章　待ち行列シミュレーション

3.1　待ち行列理論

待ち行列理論[3)]は、サービスシステムにおいてサービスを行う人あるいは機械(サービス窓口)が有限個である場合に、サービス待ちの個体数（待ち行列長さ）の変動やシステムの運転効率などについて議論するための理論である（図 3.1）。

サービスを受ける側にとっては待ち時間（システムに到着してからサービス窓口に入るまでの時間）が短いほうがよいのは当然であるし、供給側から見てもサービスした人数が利益に換算される場合には回転率を上げたいであろう。それには、サービス窓口の数を増やすのが手っ取り早い解決策であるが、人件費、設備費等を考えるとむやみにそれを増やすのは得策ではない。そこで、サービスを受ける側がストレスを感ぜず、しかも供給側には利益が上がるように“ほどほど”の人員、設備を配置することが問題となる。この“ほどほど”に対して指針を与えるのが待ち行列理論である。

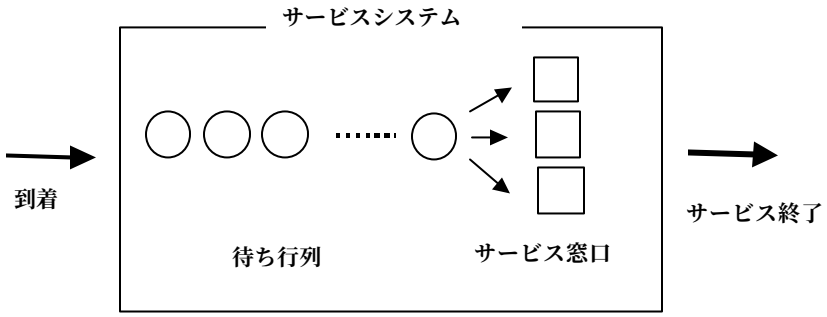

図 3.1 サービスの流れ

待ち行列の長さは客のシステムへの到着時間間隔、サービス窓口の数それから各窓口のサービス時間で決定される。到着時間間隔やサービス時間はオートメーション作業のようにきっちりと決められている場合もあるが、一般には分布を持つ。もちろん、待ち行列の長さはこの分布にも依存するが、ここでは典型的な分布である指数分布とアーラン分布について考えることにする。前述の到着時間間隔やサービス時間が決まっている場合をレギュラー分布ということもある。ここでそれぞれの分布の分布関数を与えておく。F(x)を時刻 x までに客の到着する確率とすると、

$$\text{レギュラー分布：}\ F(x)=\begin{cases}1 & (x\geq 1/\lambda)\\ 0 & (x<1/\lambda)\end{cases} \tag{3.1}$$

$$\text{指数分布：}\ F(x)=\begin{cases}1-e^{-\lambda x} & (x\geq 0)\\ 0 & (x<0)\end{cases} \tag{3.2}$$

$$\text{k次アーラン分布：}\ F(x)=\begin{cases}\displaystyle\int_0^x \frac{(\lambda k)^k t^{k-1}}{(k-1)!}e^{-\lambda k t}dt & (x\geq 0)\\ 0 & (x<0)\end{cases} \tag{3.3}$$

ただし、λは単位時間あたりの到着数（サービス数）である。

待ち行列理論を分類するためにケンドールの記号 X/Y/s(N)を導入しておく。ここで、X は到着時間間隔の分布を、Y はサービス時間の分布をそれぞれあらわし、X（あるいは Y）= D のときレギュラー分布、M のとき指数分布、E_k のとき k 次のアーラン分布をそれぞれあらわす。また、s はサービス窓口の数で、N はシステムへの入場制限をあらわす。例えば、M/M/3(∞) は到着時間間隔が指数分布でシステムへの入場数に制限はなく、窓口 3 つでサービス時間が指数分布であるような待ち行列理論をあらわす、という具合である。

（Ⅰ）M/M/c(N)　到着時間間隔、サービス時間ともに指数分布でサービス窓口の数が c、システムへの入場数の制限が N であるような場合を考える。これは、システムへの到着やサービス

終了がまったくランダムに起こる場合に相当する。このことからランダム到着、ランダムサービスの問題と呼ばれる。またこの場合、単位時間あたりの到着数やサービス数はポアソン分布に従うのでポアソン到着と呼ばれることもある。

さて、このシステムの平衡状態を考えよう。単位時間あたりの平均到着数をλ、平均サービス数をμとする。このとき、平均到着時間間隔は 1/λ、平均サービス時間は 1/μとなる。システム内にn人の客がいる確率を p_n とすると次のような漸化式が成立する。

$$
\begin{aligned}
&-\lambda p_0 + \mu p_1 = 0 \\
&\lambda p_{n-1} - (\lambda + n\mu) p_n + (n+1)\mu p_{n+1} = 0 \quad (0 < n < c) \\
&\lambda p_{n-1} - (\lambda + c\mu) p_n + c\mu p_{n+1} = 0 \qquad (c \le n < N) \\
&\lambda p_{N-1} - c\mu p_N = 0
\end{aligned}
\tag{3.4}
$$

これらを解いて、

$$
\begin{aligned}
&p_n = \frac{(c\rho)^n}{n!} p_0 \quad (0 \le n \le c) \\
&p_n = \frac{c^c}{c!} \rho^n p_0 \quad (c \le n \le N)
\end{aligned}
\tag{3.5}
$$

を得る。ここで、 $\rho = \dfrac{\lambda}{c\mu}$ と置いた。平衡状態は $\rho < 1$ の場合にのみ存在する。 p_0 は規格化条件 $\sum_{n=0}^{N} p_n = 1$ より次のように決定される。

$$
p_0 = \frac{1-\rho}{(1-\rho)\sum_{n=0}^{c} \dfrac{(c\rho)^n}{n!} + \dfrac{(c\rho)^c}{c!}\rho - \dfrac{(c\rho)^c}{c!}\rho^{N-c+1}}
\tag{3.6}
$$

式（3.5）、（3.6）よりシステムの諸量を計算することが出来る。例えば、待ち行列の平均長さ L_q とシステム平均滞在者数 L はそれぞれ、

$$
L_q = \sum_{n=c+1}^{N} (n-c) p_n = \frac{p_0 \rho (c\rho)^c}{c!(1-\rho)^2}\left\{1 - (N-c+1)\rho^{N-c} + (N-c)\rho^{N-c+1}\right\}
\tag{3.7}
$$

$$
L = \sum_{n=1}^{N} n p_n = L_q + c\rho - \frac{p_0 c (c\rho)^c}{c!}\rho^{N-c+1}
\tag{3.8}
$$

となる。また、システムに入ってからサービスを受けるまでの待ち時間の平均値 W_q、システム内の平均滞在時間 W は L_q、L と次のような関係式が成り立つ。

$$
W_q = \frac{L_q}{\lambda}
\tag{3.9}
$$

$$W = \frac{L}{l} \tag{3.10}$$

（Ⅱ）M/M/c(∞)　待ち行列の長さに制限のない場合は、（Ⅰ）の結果から N→∞　の極限を取ることにより得られ、

$$L_q = \sum_{n=c+1}^{\infty} (n-c)\, p_n = \frac{p_0 \boldsymbol{r} (c\boldsymbol{r})^c}{c!(1-\boldsymbol{r})^2} \tag{3.11}$$

$$L = \sum_{n=1}^{\infty} n p_n = L_q + c\boldsymbol{r} \tag{3.12}$$

となる。ただし、

$$p_0 = \frac{1-\boldsymbol{r}}{(1-\boldsymbol{r})\sum_{n=0}^{c} \frac{(c\boldsymbol{r})^n}{n!} + \frac{(c\boldsymbol{r})^c}{c!}\boldsymbol{r}} \tag{3.13}$$

である。

（Ⅲ）M/E_k/1(∞)　ポアソン到着で、k次のアーランサービス、窓口 1 つで待ち行列に制限のない場合を考える。この場合は、サービスがk段階に別れていて、各段階のサービスが平均サービス時間 1/k μの指数分布であるようなものを考えれば良い。もちろん同時に一人しかサービスを受けることは出来ない。k = 1 の場合には指数サービスに一致する。結果のみ示すと、平均待ち行列長さ L_q、平均滞在者数 L はそれぞれ、

$$L_q = \frac{\boldsymbol{r}^2}{2(1-\boldsymbol{r})}\left(1+\frac{1}{k}\right) \tag{3.14}$$

$$L = L_q + \boldsymbol{r} \tag{3.15}$$

となる。また、この場合も関係式（3.9）（3.10）が成り立つ。

3.2 シミュレーション

シミュレーションは、単位時間を適当な数に分割し、離散時間での待ち行列の時間発展を数値的に追う。図 3.2 は計算条件の入力画面である。

入力する条件は、単位時間あたりの客の到着数、サービス数、到着分布とサービス分布、初期行列の長さ、サービス窓口の数、行列長さの制限、実行時間、実行回数、単位時間の分割数である。到着分布は（レギュラー、ポアソン（指数））分布から、サービス分布は（レギュラー、ポアソン（指数）、アーラン）分布から選択できるようにした。レギュラー分布では客の到着時間間隔やサービス時間は一定であるのに対して、それ以外の分布では現象は確率に支配される。ここでは、Visual BASIC の組み込み関数 Rnd を使用して単精度の乱数を発生させた。

ここで、プログラムの中で待ち行列の長さ lq の時間発展を記述している部分を見ておくことにする。

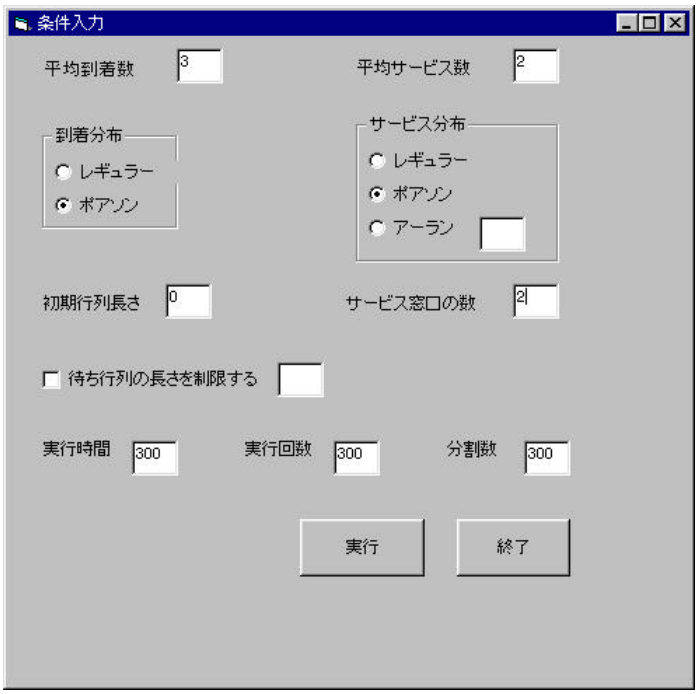

図 3.2 計算条件の入力画面

(a) **ポアソン到着の場合には、p＝λ/（単位時間の分割数）として、各離散時間において、次のように増分を行う。**

```
If Rnd < p then
    If（すべての窓口がサービス中　かつ　lq < 行列長さの制限）Then
        lq = lq +1
    Else
        I=0
        Do While I <（窓口の数）
            I=I+1
            If Flag(I) = 0 Then
                Flag(I) = 1
                Exit Do
            End If
        Loop
    End If
End If
```

ここで、Flag(I)=1(0)**は** I **番目の窓口がサービス中（空いている）ことをあらわす。**

(b) **指数サービスの場合には、q＝μ/（単位時間の分割数）として次のように** lq **の減分を行う。**

```
For I = 1 To（窓口の数）
    If Rnd < q Then
        If lq > 0 Then
```

```
            lq = lq –1
            Flag(I) = 1
        Else
            Flag(I) = 0
        End If
    End If
Next I
```

(c) k次アーランサービスの場合には、q＝k μ/（単位時間の分割数）として、

```
For I = 1 To（窓口の数）
    If Rnd < q Then
        If lq > 0 Then
            If Phase(I) = k 0r Phase(I) = 0 Then
                lq = lq –1
                Phase(I) = 1
            Else
                Phase(I) = Phase(I) + 1
            End If
            Flag(I) = 1
        Else
            If Phase(I) = k Then
                Phase(I) = 0
                Flag(I) = 0
            Else
                If Phase(I) <> 0 Then
                    Phase(I) = Phase(I) +1
                    Flag(I) = 1
                End If
            End If
        End If
    End If
Next I
```

のように lq の減分を行う。ここで Phase(I)は I 番目の窓口が何段目のサービスを行っているかをあらわす量である。

以上は待ち行列長さの増減であるが、システム内の滞在者数は待ち行列の長さにサービス中の窓口の数を足すことにより得られる。また、サービスまでの待ち時間やシステムへの滞在時間は、実行時間内で最後にシステムに入った客の到着時間とサービス開始時間、サービスを終えてシステムを離れた時間を記録することにより得られる。

結果の出力は、3 通りの形式で行う。図 3.3 は待ち行列長さの変動を第一回目の試行に関し表示したものである。レギュラー到着でレギュラーサービスの場合はこの出力のみである（図 3.4）。図 3.5 は各離散時間での待ち行列長さの平均をプロットしたものである。これにより、平衡解への接近（過渡解）の様子が分かる。図 3.6 はテキスト形式での結果出力である。ここでは、計算条件とともに、平均行列長さ、平均滞在数 、平均待ち時間 、平均滞在時間の各シミュレーション結果が表示される。解析的な結果が得られている場合にはあわせてそれも表示

ュレーション結果が表示される。解析的な結果が得られている場合にはあわせてそれも表示するようにした。

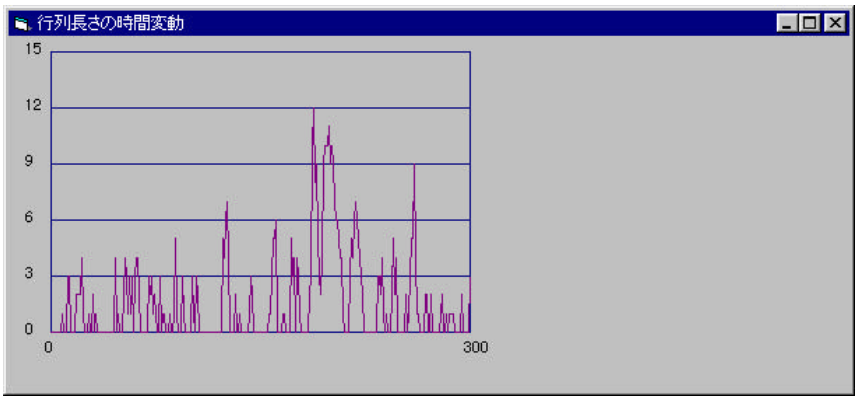

図 3.3 行列長さの時間変動（ポアソン到着－指数サービス）

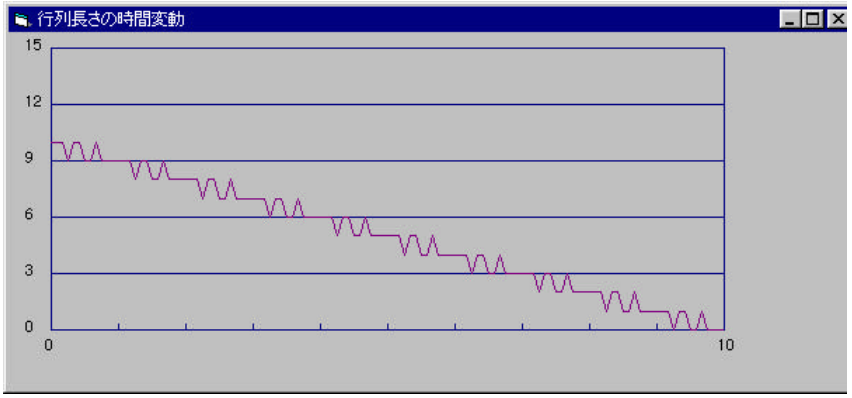

図 3.4 行列長さの時間変動（レギュラー到着－レギュラーサービス）

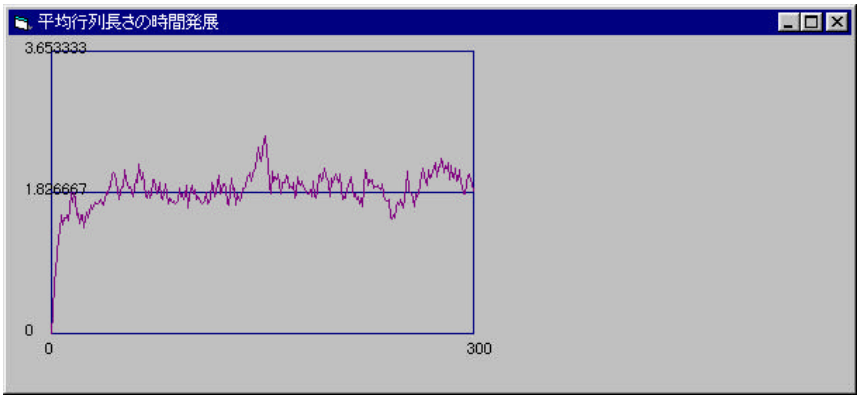

図 3.5 平均行列長さの時間発展

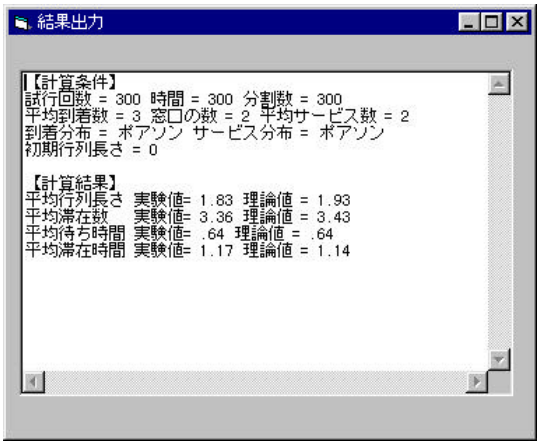

図 3.5 計算結果の出力画面

４章　考察と今後の課題

線形計画法の実用パッケージを作るには、退化問題、変数や制約式の数が増えた場合のメモリの有効活用問題、誤差処理の問題等、様々な壁がある。これらの状況を考えると、今のプログラムの段階は初歩的である。しかし、現段階である程度の規模の問題に対して実用になるものを作ることは、プログラムの枠組みが出来あがっているのでそれほど難しいことではない。それ以上に巨大な問題については、専門家の領域であり、素直に専門のソフトウェアに任せればよい。我々の目指すソフトウェアは、主として学生の学習用であり、社会へ出て専門家になるまでの橋渡しの期間に利用されるものである。以下、どのような点を改良すれば、実用になるかを考える。

まず最初に考えなければならないものとしては、改訂シンプレックス法の導入である。通常のシンプレックス法はメモリを多く消費し、誤差の蓄積も無視出来ない。実務的にも教育的にも改訂シンプレックス法の導入は不可欠である。但し、これ自体は特に難しい問題ではないが、教育用としてどのように視覚化するかは今後の課題である。

次に、退化の処理も理論的には重要である。実際上、退化はあまり起きることはないと言われているが、その対策は組み込んでおくべきであろう。またデータ形式として、数式表現からの入力も考える必要がある。変数や制約式の数が増えると表形式での入力は煩雑になり、見たままの数式表現からの入力も望まれる。

今後、線形計画法だけでなく数理計画法一般に対してもソフトウェアを追加して行く予定である。教育における重要性と緊急性を考えてプログラム化する手法を選びたい。

待ち行列シミュレーションについて、解析的な結果が得られている場合の計算結果を理論値と比較すると、実行回数が 300 回では誤差がほぼ１０％以内に抑えられている。これは、シス

テムが平衡に達したと見なされる状態での揺らぎ幅とほぼ一致する。これは、実行回数を増やせばある程度まで小さくなり、１０００回ではほぼ 5%以内に抑えられている。ただし、この調子で精度を上げることが出来るかというとそうではなくて、これ以上実行回数を稼いでも、結果に改善があまり見られない。これは、乱数の一様性と関連しているものと考えられる。実際、乱数が一様であると仮定するとイベントの生起回数はポアソン分布になるはずであるが、試行回数を増やしてもこれに近づかない。従って、Visual BASIC の組み込み関数 Rnd を用いる限りはこのあたりの精度が限界であろう。これは、別のアルゴリズムの乱数を用いて確認してみたい。また、到着分布やサービス分布は一様分布から得られる指数分布やアーラン分布を用いたが、任意の分布形状を入力できると便利かもしれない。例えば、実際に店頭等で観測を行って分布を調べ、その結果を入力して、サービスの改善などに役立てることが出来れば面白い。さらに、到着数の時間変動を入力し、それによる待ち行列長さの応答を調べるなどすれば現実的な問題に近づくであろう。これらは今後の課題である。

参考文献

1) 福井正康, 田口賢士, 社会システム分析のための統合化プログラム, 福山平成大学経営情報研究, 3 号, 109-127, 1998.
2) 福井正康, 田口賢士, 社会システム分析のための統合化プログラム２　－産業連関分析・KSIM・AHP－, 福山平成大学経営情報研究, 3 号, 129-144, 1998.
3) 三根 久, オペレーションズ・リサーチ（上巻・下巻）, 朝倉書店, 1976.
4) 刀根薫, 数理計画, 朝倉書店, 1978.
5) オペレーションズリサーチⅠ,Ⅱ, 森雅夫, 森戸晋, 鈴木久敏, 山本芳嗣, 朝倉書店, 1991.
6) L.Schrage, 新村秀一・高森寛訳, 実践数理計画法 －LINDO を用いて－, 朝倉書店, 1992.

Multi-purpose Program for Social System Analysis 3

- Linear Programming, Queuing Theory -

Jun-ichi MASKAWA, Masayasu FUKUI and Katashi TAGUCHI

Department of Management Information, Faculty of Management,
Fukuyama Heisei University

Abstract

The authors present a detail explanation of theory and program for linear programming and queuing simulation in the way of constructing unified software on social system analysis. The aim of this program is to grasp a theoretical idea of the simplex method for the linear programming and to understand the theoretical analysis and the simulation method for queuing theory.

Keywords

linear programming, simplex method, queuing theory, simulation, software, unified program