# ビジネスのための数理科学

田辺　隆人（NTTデータ数理システム）[1]

## １．はじめに

我々はビジネスにおける物や人の流れをより詳細に知るため、現実の一部を数理モデルの形で切り取って連続あるいは離散系のシミュレーションを使って解析する。数理最適化を用いれば、観測との違いを最小化するような、あるいはビジネス上望ましい特性を最大化するような、数理モデルのパラメータを探索することもできる。潤沢にデータがあれば、必ずしも数理モデルが最初からわかっていなくてもよい。いくつか試して当てはまりのよい数理モデルを帰納的に選択することもできるし、究極の柔軟性を備えた多層ニューラルネットワークに原因と結果だけを与えてパラメータを最適化し、その結果を使って人間の判断の一部を肩代わりさせることもできる。

計算機の進歩によって数理科学の適用分野は大きく広がった。その成果は我々の日常に入り込み、我々は時代の「進歩」を実感することができる。ただ、数理科学が解いている問題・解析手法の大半は古典的なものが多い。例えば、我々が非線形方程式を解く場合にまず例外なく頼るNewton法、一次方程式を解く際に用いるGaussの消去法から派生する様々な行列分解の手法は言うには及ばず、もっとも広く使われている最適化手法である線形計画法の理論を基礎付けるFarkasの補題や非線形最適化の最適性の必要条件を表現したkarush-kuhn-tucker条件もそれらが確立されて数十年が経過している。Google創業者のチームは、ページ遷移確率行列の固有ベクトルを求めることでページランク（の原型）を求めたがその際に用いたのはベクトルを正規化しながら遷移確率行列を掛け算することを繰り返す古典的な方法（power method）であった[1]。

理論と応用の発展にギャップがあるのは特段珍しいことではなく、解いている問題や手法が古典的だからといって応用の価値が下がることは決してない。また、我々数理科学を実践する立場の人間は、特にビジネス上の価値観（実現できること、高速に動作すること）を実現しようとする中、細かな点においてかなりしばしばよく解析されていないと思われる（少なくとも教科書には載っていない）事象に行き当たり、それを解決する必要に迫られる。その幾つかは数学コミュニティーから得られる情報を用いることで見通しが得られて、我々は多くの局面で救われている。が、事象の中には完全には未解決なものも多く、恐る恐る仕事をせざるを得ない場合もある。本稿ではそんな知り尽くされているはずの古典的な問題を具体的に解こうとするときに立ち現れる難しさについて述べてみたい。

[1] tanabe@msi.co.jp

## ２．疎行列

熱伝導などの物理現象を記述する拡散方程式：

$$\nabla^2\phi = \rho \qquad (1)$$

を計算機上で解こうとするとき、物理量を離散点上に対応付け、何等かの規則（差分スキーム）で偏微分方程式を離散点同士の関係に変換することが行われる。例えば差分スキームとして有限体積法を用いると（１）は、離散点上の物理量$\phi_i$同士の関係として

$$\sum_{j\in N_i} a_{ij}\cdot\phi_j - a_{ii}\cdot\phi_i = \rho_i \qquad (2)$$

と表現できる。ここで$a_{ij},\rho_i$はメッシュ形状から現れる定数、$N_i$は点$i$に近接している点の集合である。

数理最適化で「割り当て問題」と呼ばれるクラスの問題は非常に応用範囲が広く、例えば学校と学区の対応、救急車両と車両が担当する領域の対応、インターンと病院の対応などを、全体を鑑みてバランス良く行うことに貢献する。割り当て問題の変数は、割り当てられる物の添え字 $i \in N$（学区、地域、インターン）と割り当て先を表す添え字 $j \in M$（学校、救急車両、病院）の組で添え字づけられる 0−1 変数 $x_{ij} \in \{0,1\}$であり、割り当ての有無を示す。可能な対応付け全体の集合$IJ$を用いて

$$\sum_{j,(i,j)\in IJ} x_{ij} = a_i$$
$$\sum_{i,(i,j)\in IJ} x_{ij} = b_j \qquad (3)$$

という式が問題の構造を示す。ここで、$a_i, b_j$は対応付けの上限数である。

Google ページランクは、ページ$i$を見ていたユーザーが次にページ$j$に推移する確率$p_{ij}$を要素として持つような行列$P$の固有ベクトルを求めることで定められた。遷移確率の定義から、行列$P$の要素は

$$\sum_{j,(i,j)\in IJ} p_{ij} = 1 \qquad (4)$$

という関係を満たす。ここでの集合$IJ$は遷移を行うインデクスの組の集合である。

（２）、（３）、（４）の線形性に着目すれば、これらを適当な行列とベクトルへの読み替えを用いて

$$Ax = b \qquad (5)$$

と表現することが可能である。ただ、行列$A$を計算機上に展開するには戦略が必要となる。例えば拡散方程式のケースに対して三次元の解析領域を 100 分割した場合を想定すると変数の個数は$10^6$個のオーダー、すなわち（２）を表現する行列の次元は$10^6$のオーダーとなる。従って$A$の全要素をそのまま蓄えるにはその二乗、$10^{12}$のオーダーの領域が必要となり現実の計算機で簡便に扱うことができる大きさを大幅に超えてしまう。（４）の行列の次元はインターネットの全ページ数（[1]によると 1990 年代の後半でも$10^8$を超えるオーダーだったという）であり、やはり二乗のオーダーの領域は準備できない。（３）の行列も（割り当てられる物の数×割り当て先の数）の次元を持っていて、あるサービスのユーザー（例えば数万）に対する店舗（数百）

レコメンドなどを考えると、次元数が$10^6$を超える場合も珍しくない。
こうした場合にほぼ例外なく取られるのが行列を「疎行列」として保持する、すなわち要素が零でない部分のみを計算機上に保持する、という方法である。より具体的には行列をその次元と、値の羅列のみで表現するのではなく、「どの場所に非零があるのか」という情報（「行列構造」と称されることが多い）を用いて表現するということである。この効果は絶大で、行列を表現する上で必要な領域、操作に必要な時間が大幅に節約できる。（２）であれば、行列の非零要素は変数の高々定数倍、（３）であれば行列の非零要素数は行列の次元数の二倍、（４）の行列の非零要素数はウェブページのリンクの数と同じなので、なんとか現実の計算機が扱うことのできる領域の中に納めることができる。計算速度の観点で見ても、疎行列とベクトルの積は非零要素に比例する数の計算操作で実施できるので、この表現形式を取ることで計算効率も同時に向上させることができる。具体的には（２）の拡散方程式系を解くのには行列とベクトルの積のみで実行できる一次方程式解法の一種である反復法を使えば良い。（４）で固有ベクトルを求める際に用いた power method も素朴な方法ではあるが、やはり行列とベクトルの積に帰着するので、計算の実行可能性を考えると適切な算法の選択であったと言える。

## ３．一次方程式と疎行列

前項で言及した反復法は、一次方程式の解が行列を使って表現された多項式の最小点と見做せることを用いた一次方程式解法である。従って演算そのものは最適化アルゴリズムの実行に等価であり、最適化の過程において行列とベクトルの積のみしか必要としないことが計算効率に大きく貢献している。しかし現実には左辺行列の条件数次第で最小化がうまくゆかず十分な精度が確保できない場合もある。そういう場合にはガウスの消去法から導かれた直接法を用いる。

行列の一次方程式の直接解法とは左辺行列$A$を対角要素が１の下三角行列$L$と対角行列$D$と上三角行列$U$の積

$$A = LDU \qquad (6)$$

として表現してから（行列の分解と呼ばれる）、三角行列に関する一次方程式が代入操作のみで解けることを利用して解を導く解法である。直接法を効率よく実行するためには$L$、$U$が疎行列である必要があるが、$A$が疎行列でも$L$、$U$が疎行列であるとは限らないので、$A$の行・列の並べ替え（fill-reducing ordering）を行って$L$、$U$の疎行列性を確保する。拡散方程式系を解く場合や数理最適化アルゴリズムの一種である内点法を適用する場合など、左辺行列$A$が対称である場合が応用で頻出するが、その場合には変数と、変数が対角要素に現れている式を同一視してグラフ$G(N,E)$のノード、行列の非零要素を変数と式を結ぶリンクにそれぞれ対応させて考えると、$A$の分解操作が$G(N,E)$の操作として見通し良く記述できる。その描像の下で fill-reducing ordering は $G(N,E)$のノードを消去する順番を決める問題であると定義することができ、この表現を用いて minimum-degree アルゴリズム[6]など、有用なヒューリスティクスが多数提案されている。

ただ、行列の分解結果が疎行列であることは、分解の計算操作の高速化と表現効率の向上にはつながるものの、計算精度については何も担保していないことに注意しなければならない。$A$の分解操作は$A \equiv \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$と表現して右下の部分行列を

$$A_{22} \leftarrow A_{22} - A_{21}A_{11}^{-1}A_{12} \qquad (7)$$

と更新することに帰着するが、対角要素$A_{11}$のノルムが零に近い場合には（7）の更新の際に行う加減算で絶対値の差の激しい数同士の演算を行うことによる情報落ちが起きやすくなり、結果の精度が落ちてしまう。そのため、特に行列の条件があらかじめ担保できない非線形最適化から現れる行列では計算速度と精度のトレードオフを狙って fill-reducing ordering に従いつつ、行列の分解の途中で対角要素の値を見ながら行列の列・行の並べ替え（pivoting）を行う方法[7]が提案されている。

行列操作は数理モデルの解析アルゴリズムの実行において大きな計算機資源を消費する部分であるため、ハードウエアの特性を利用した工夫も重要である。行列演算の核心部が読み書きするデータを、高速に読み書き可能なキャッシュメモリに収め、間接参照が含まれない形にするという工夫は現在普及しているコンピュータを高速に動作させるために極めて有効な工夫である。ただ、疎行列は非零要素の場所と値をそれぞれデータとして持つ方式なので、その操作において間接参照を排除しにくい。そこで非零要素のパターンが同一な列をまとめて「スーパーノード」とみなして間接参照を排除する、あるいは少々の零要素を非零扱いして間接参照を減らす（node amalgamation)、といった細かな実装の工夫が行われる。ただ、容易に想像できるようにこの工夫は上述したような行列の分解の途中で行列の列・行の並べ替えを行う方法と親和性が悪い。従って前述した計算速度と精度のトレードオフを狙ってピボット選択を行う実装では、行列分解の核心部が間接参照なしで行われる multi-frontal 法と呼ばれる行列分解アルゴリズムが利用されている。

ここまで見てきたように一次方程式の直接法は古典的な手法であり、手続きとして明確に定義されているが、数理科学の応用例から現れる問題を計算機上で解くとなると、上に紹介したような手法を組み合わせて精度や速度を実現することが不可欠となる。ここでは主に直接法について紹介したが、反復法についても条件が悪い一次方程式を御するためのスキーム（性質の悪い行列を使って表現された多項式の最小化を安定に行うための技法）や行列の性質を改善するための前処理方法が数多く提案されている。なかでも前処理法の一つとして直接法を部分的に適用して「大雑把」に行列を解く不完全 LU 分解を伴う反復法では、反復法と直接法のノウハウを融合して適用する必要がある。

## ４．自動微分

数理科学の多くのタスクは計算機上に数学的な関数を表現し、関数を最小化する点を求める、あるいは特定のパラメータに対する感度（微係数）を調べることに帰着する。計算機上での関数 $\mathrm{F}: \mathrm{R}^n \to \mathrm{R}^m$の表現形に着目して関数の微係数を求める方法が自動微分法という方法である。計算機上に数学的な関数を表現するには、プログラミング言語が備えている単項演算（べき乗や指数関数、対数、三角関数など）あるいは二項演算（加減乗除）を組み合わせて用いるのが普通である。さらに複雑な関数は、その定義を入れ子にして表現する。例えば、$n$次の多項式$f(x) = \sum_{i=0}^{n} a_i x^i$を表現するには$f_0 \leftarrow a_n$とした後に

$$f_i \leftarrow x \cdot f_{i-1} + a_{n-i} \qquad (i = 1 \cdots n) \tag{８}$$

という漸化式を手続き的に計算すればよい（Horner's method)。

関数$\mathrm{F}: \mathrm{R}^n \to \mathrm{R}^m$がこのように手続き的に定義されているとき、具体的にはFを定義する手続きの各ステップが中間変数 $\mathrm{u}_i (i = 0 \cdots p) \in \mathrm{R}^{s_i}$と関数$\mathrm{F}_i: \mathrm{R}^{s_{i-1}} \to \mathrm{R}^{s_i}$を用いて

$$\mathrm{u}_i = \mathrm{F}_i(\mathrm{u}_{i-1}) \qquad (i = 1 \cdots p) \tag{９}$$

のように定義されているとき（ただし$\mathrm{u}_0 \equiv \mathrm{x}$、$\mathrm{F} \equiv \mathrm{u}_p$と約束する)、chain-rule を用いるとFのヤコビ行列 $\mathrm{J} \in \mathrm{R}^{m \times n}$は$\mathrm{F}_i$のヤコビ行列$\mathrm{J}_i \in \mathrm{R}^{s_i \times s_{i-1}}$の行列の積

$$\mathrm{J} = \mathrm{J}_p \cdot \mathrm{J}_{p-1} \cdot \cdots \cdot \mathrm{J}_2 \cdot \mathrm{J}_1 \tag{１０}$$

として記述できる。ここで各計算ステップのヤコビ行列$\mathrm{J}_i$（要素導関数）の要素の計算は、$\mathrm{F}_i(\mathrm{u}_{i-1})$の計算と同程度の手間で可能であることに注意されたい。ただ、実際の計算の効率を追求する場合、(１０)の結合順序に注意しなければならない。もし、Fがスカラー関数であれば、$\mathrm{J}_p = \mathrm{J}$は行ベクトルになるので、（１０）は左から結合してゆくのが計算速度として有利になる。この微係数の計算手法は高速自動微分法、reverse automatic differentiation また、多層ニューラルネットのパラメータチューニング（ロス関数の最小化）の文脈においては back propagation[8]と呼ばれている。しかし、この手法は、計算の各ステップにおいて計算されてゆく$\mathrm{J}_i\ (i = 1 \cdots p)$をFの最終結果が出るまで保持しておかねばならない点で、計算に必要な作業領域が大きくなる。この問題を解消するため、一部の$\mathrm{J}_i$を再計算することを許して作業領域を節減する方法（check-pointing）が提案されている。また（１０）の計算における計算の効率性を追求するにあたっては、行列の積の結合順序は任意であることを利用してさらに演算の手間が削減できる可能性があることもあわせて指摘しておきたい。計算の手間を最小化すべく行列の結合順を決定する問題は$\mathrm{J}_i$が疎行列でなければ行列積の手間が行列のサイズのみで表現できるので、動的計画法により効率よく解くことができる。しかし、一般に$\mathrm{J}_i$は疎行列になり、疎行列同士の積のコスト、積の結果現れた行列の構造は積のオペランドになっている疎行列の構造そのものから決定するので演算の手間の削減はあまり簡単ではない。筆者はモデリング言語によって表現された関数の一階・二階微係数の計算を、行列積の結合順序と疎行列性を同時に考慮するヒューリスティクスによって高速化、結果として内点法による非線形最適化の効率を向上させられることを示した[5]。

## ５．混合整数計画問題

数理最適化問題

$$\text{最小化 } f(x)$$
$$\text{制約 } g(x)=0,\ x\in[l,u] \qquad (11)$$

において一部の変数が整数でなければならないという制約の付いた問題を混合整数計画問題と呼ぶ。この問題は実務的・政策的な意思決定を行う上で重要であり、特に$f(x), g(x)$が線形関数である問題を解くアルゴリズムは広く研究されている。混合整数計画問題は代数的手続きによって直接解くことができない。例えば、

$$x+5y+4z=10$$
$$4x+y-2z=7 \qquad (12)$$

$x, y, z$の組で $|x-y|$ を最小化するものを求めようとするとき、整数性の制約がなければ、$x-y=0$を連立させて代数的手続きを適用するのみで事足りるが、$x, y$ に整数性の条件を課すと $x-y=0,\pm1,\pm2,\cdots$ という場合分けを行い、整数解が出るまで代数的手続きを繰り返す必要がある（答えは$x-y=-8$ としたときに見つかる $x=-2, y=6, \mathrm{z}=\frac{1}{4}$)。さらに制約式の係数が

$$19x+5y+19z=23$$
$$17x+y-2z=29 \qquad (13)$$

と変化すると、整数性の条件がない場合の結果（緩和解と呼ばれる）と整数解はさらに隔たって、$x-y=87$としたときに初めて整数解が現れる。混合整数計画問題の汎用解法である分枝限定法はシステマティックな場合分けによって最適解の存在しない領域を消去（枝刈り）し、最適解を求める方法であるが、整数解は発見的な方法に頼るので、いかに早く最適解が求まるかは緩和解（整数性の条件を緩めた解）と真の解の隔たりによって大きく左右される。上で見たように、それは変数や制約の数といった外面的特徴のみから計測することは難しく、実務において最適化を適用する場合の難しさの一つとなっている。手掛かりがないわけではない。次は統計的マッチングといった文脈で今も解かれているHitchcock-Koopmans輸送問題と呼ばれる古典的な線形計画問題である。

$$\text{変数} \quad x_{ij}\geq 0$$
$$\text{最小化} \quad \sum_{(i,j),(i,j)\in IJ} c_{ij}x_{ij}$$
$$\text{制約} \quad \sum_{j,(i,j)\in IJ} x_{ij}\leq s_i$$
$$\sum_{i,(i,j)\in IJ} x_{ij}=d_j \qquad (14)$$

ここで変数$x_{ij}$は供給地から需要地に運ぶ輸送量を表していて、供給地には供給上限$s_i$が、需要地には需要量$d_i$が定義されている。集合$IJ$は供給地と需要地の組で表した輸送経路の集合で、$c_{ij}$は各輸送経路のコストである。実はこの問題は線形制約を定義する行列が total unimodularity という性質を持つため、$d_i$が整数であるとき、線形計画法の解法に単体法を用いるならば、必ず整数の最適解が得られることが知られているため、整数解を求めるにあたっての困難はない。

この問題も輸送量に上限制約を加えるだけで、左辺行列の total unimodularity

が壊れてしまうので、実務的な場合に分枝限定法が全く不要というわけではないが、この問題から派生した数理最適化問題は整数解と緩和解の隔たりは小さく、効率良く解ける。実務家としてはこの輸送問題のケースのようになるべく緩和解と整数解の隔たりが少なくなるような（strong な、と言われる）定式化を探したい。

[2]では上の輸送問題に供給地を閉鎖するかどうかという決定$y_i \in \{0,1\}$を加えて、輸送量を$1 \geq x_{ij}$と制約し、$d_j = 1$と設定した問題（施設配置問題）について考察している。そこでは「閉鎖された供給地からの輸送量は零」という意味の定式化を

$$x_{ij} \leq y_i \qquad （１５）$$

と表現する場合と

$$\sum_{j,(i,j)\in IJ} x_{ij} \leq s_i \cdot y_i \qquad （１６）$$

と表現する場合の違いを精密に論じている。$y_i \in \{0,1\}$であれば（１５）、（１６）は等価であるが、緩和解を考えるとそうではなく、（１６）は（１５）を集積した形になっていることから、（１５）の表現を用いた方が整数解と緩和解の隔たりが少なくなり、分枝限定法による解法は効率化される。このように整数性を考慮した場合に冗長で規模の大きな表現が、実は strong な定式となっていて、計算が効率化されることがある。もちろんいつもそうとは限らない。線形な混合整数計画の重要なテクニックとして、整数性の制約を満たす範囲で緩和解の実行可能領域（制約を満たす領域）を削減する式（切除平面）を自動的に導く手法が知られているが、切除平面を沢山加えすぎると今度は連続緩和された問題の規模が増大して解きにくくなるので、必要十分な切除平面を選別する手法（cut selection）が重要になる。これまで提案されている切除平面法と cut selection にまつわる問題については[4]のレビューが詳しい。

現在の混合整数計画法については切除平面法など様々なテクニックが提案されており、最適化ソルバーはそれらを独自の方法で組み合わせて実装されている。このような状況では特定の定式化の方法が、ポピュラーな最適化ソルバーに対してどの程度親和的か（早期に結果を出すか）もノウハウとして重要である。[3]ではジョブショップスケジューリング問題という古典的な整数計画問題に対して、これまでに提案された意味上等価な定式化を、様々な問題インスタンスに対して適用し、複数の数理最適化ソルバーがどの程度速く解けるかを示している。

## ６．おわりに

数理科学の多くのビジネス上の応用部面において現れる数理モデル化や解析の方法は、ガウスの消去法から派生した行列分解や chain-rule から派生した自動微分法、線形計画問題を解く単体法、などいずれも古典的で見通しのよい手法に基づいている。ただ、現実の計算機を使ってそれらをいざ実行させようとするとその途端に「実現可能性」、「速度」そして「安定性」といった要求が現れて、実装のための先人の知恵やアルゴリズム、理論的知見が不可欠であることに気づかされる。疎行列の分解がグラフ上の操作として理解できたり、自動微分の手法が行列の積の順番の発見という形で整理できたり、混合整数計画問題の解きやすさが行列の性質に結び付い

たり、現実世界の要求を追求してゆく中で、教科書的には「知り尽くされた」はずの話がいろいろな方向に発展してゆく様子が筆者にはたまらなく面白く豊かなものに感じられる。

現実の問題の解決において自分には馴染みのない問題に直面したとしても今や一人きりで悩むことはない。筆者が数理科学にかかわり始めた約３０年前と比べると情報流通のスピード、質、量は劇的に改善し、インターネットで検索した先人の業績を適用することで数十年の発展の歴史を短期間で享受して結果を大幅に改善することも可能である。ただ、どうしてもうまくゆかない問題に行きあたってしまう場合も多い。理由をよく考えてみると、自分が相手にしている問題は先人が取り組んだのとは完全に同一ではなく、設定や目的において微妙にずれていることに起因することが多い。そう認識したところから我々の孤独な探索は始まる。目の前に提示されるのは現実世界の一部であり、ビジネス上の要求も様々である。先人の方法をなぞるだけで解決できる保証はどこにもなく、ヒューリスティクスであれアルゴリズムのパラメータの調整であれ、何等かのオリジナルなやり方を導いて顧客の要求に応える結果を出さなければならない。そんな中で我々の道しるべとなってくれるのが数学コミュニティーのもたらしてくれる情報である。収束の定理、問題クラスの分類、アルゴリズムの計算量の上界、などなど、たとえ肯定的な結果でなくてもよい。否定的な結果でも我々のあてどない探索の道筋を照らし、袋小路に入るのを防いでくれている。そういう意味で筆者は数学の理論の世界に多くを負っている。私の立場からどんな形で数学の世界に貢献できるだろうか。まずは数理科学が現実世界をモデル化してより良い形を探る上で、数学が目立たない形でもどれほど大きく役に立っているかを語り続けてゆく所存である。

## 参考文献


[1] Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry(1999), The PageRank Citation Ranking: Bringing Order to the Web. Technical Report. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/

[2] Magnanti, T. L. & Wong, R. T.(1981), Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. Operations Research, 29, 464-484

[3] Ku, W. Y., & Beck, J. C. (2016), Mixed Integer Programming models for job shop scheduling: a computational analysis. Computers & Operations Research, 73, 165-173

[4] Dey, S.S. & Molinaro (2018), Theoretical challenges towards cutting-plane selection,M. Math. Program. 170: 237. https://doi.org/10.1007/s10107-018-1302-4

[5] 田辺隆人、数理計画法に適した自動微分算法の開発と実装、中央大学,2005,博士論文

[6] A. George and J. W. H. Liu(1989), The evolution of the minimum degree ordering algorithm, SIAM Review, 31, pp. 1-19

[7] Duff, I. S. (2004), 'MA57 – A code for the solution of sparse symmetric indefinite systems', ACM Trans. Math. Softw. 30(2), 118–144.

[8] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (1986-10-09),Learning representations by back-propagating errors,Nature. 323 (6088): 533–536.