EXCEL 表計算による Sheet 上プログラミング

―――PC 授業への展開―――

日大生産工　　○篠原正明
情報システム研究所　　篠原健

１．はじめに

EXCEL は表計算用ソフトであり、表形式で与えられたデータを効果的に処理する特徴を持つ。本来は、事務データ処理を目的としていたが、科学技術計算への応用も進んでいる。EXCEL の機能は、Sheet 上での表計算処理と VBA で書かれたアドインソフト(アドオンソフトとも言う)による応用計算処理に大別できる。Sheet 上の表計算処理では、セル毎に定義された計算式の処理が実時間で実行され(関数)、アドインソフト応用計算処理では、VBA（**Visual Basic for Applications**) のプログラムが実行される(サブルーチン)。

Sheet 上の表計算処理は、処理が迅速、処理内容が理解容易、データ入力が容易、計算結果の表示が明確、処理プロセスの accountability、等などの長所があるものの、複雑な処理には向かないため、アドインソフトを併用する問題解決アプローチを採用するのが普通である（例えば、〔１〕参照)。しかし、アドインソフトは内蔵プログラムであり、その処理自体がブラックボックスとなり、前述した Sheet 上の表計算処理のもつ多くの長所(例えば、処理内容が理解容易、計算結果の表示が明確、等)がそこなわれてしまう。そこで、Sheet 上の表計算処理のもつ多くの長所を保持した上で工学上発生する応用科学技術計算を処理する方法として、「Sheet 上プログラミング」を提案する。

２．Sheet 上プログラミング

Sheet 上プログラミングとは、アドインソフトなどのマクロ機能を使わずに、Excel Sheet 上の代入文(関数も含む)のみにより、求解手続きを与える計算法である。

３．Sheet 上プログラミングの例

3.1　多重回帰分析

アドインソフトで実現するならば、被説明変数のデータベクトル y、説明変数のデータ行列 X を入力して、回帰パラメータベクトル P を出力するサブルーチン（BLACK　BOX）となる。

一方、Sheet 上プログラミングでは、以下の手続きを Sheet 上で代入文の連なりとして入力することにより、最終的に W_4 において P が計算される。

y ＝{入力データ}

X＝{入力データ}

$\mathrm{W}_1 = \mathrm{X}^T\mathrm{X}$

$\mathrm{W}_2 = \mathrm{W}_1^{-1}$

$\mathrm{W}_3 = \mathrm{W}_2\mathrm{X}^T$

$\mathrm{W}_4 = \mathrm{W}_3 y$

On-sheet Programming via EXCEL Spreadsheet
Masaaki SHINOHARA and Ken SHINOHARA

3.2　AHP 一対比較のウェイト推定

アドインソフトで実現するならば、完全一対比較行列 A を入力して、固有ベクトル法によるウェイトベクトル x を出力するサブルーチンとなる。固有ベクトルのルーチンを使っても可能である。

一方、Sheet 上プログラミングでは、以下の手続きにより Power 法を用いて、最終的に v_{10} において x が計算される。

A ={入力データ}

x_0 ={入力データ}

$w_1 = Ax_0$

$v_1 = w_1 / |w_1|$　(w_1 の全要素総和が 1 に正規化の意味)

$w_2 = Av_1$

$v_2 = w_2 / |w_2|$

：

$w_{10} = Av_9$

$v_{10} = w_{10} / |w_{10}|$

ここで、$|w| = \sum w(i)$ はベクトル w の L_1－ノルムである、又 Power 法を反復回数＝10 で停止しているが、10 回反復すれば、収束しているだろうという予想のもとでの手続きである。

3.3　マルコフ連鎖の定常状態ベクトル計算

アドインソフトで実現するならば、遷移確率行列 P を入力して、定常状態ベクトル x を出力するサブルーチンとなる。べき乗法、あるいは連立方程式などにより求解できる。

一方、Sheet 上プログラミングでは、以下の手続きにより反復計算を用いて、最終的に W_{10} の各行で x が計算される。

P ={入力データ}

$W_1 = P \times P$

$W_2 = W_1 \times P$

$W_3 = W_2 \times P$

：

$W_{10} = W_9 \times P$

ここで、反復回数＝10 で停止しているが、不安ならば、20～30 回の反復手続きを Sheet 上に書けばよい。

3.4　最短路行列の計算

アドインソフトで実現するならば、距離行列 $A(a_{ij} : i \to j$ の距離；枝が存在しなければ＋∞)を入力して、最短路行列 X を出力するサブルーチンとなる。そのアルゴリズムとしては Warshall-Floyd 法が有名である。

一方、Sheet 上プログラミングでは、以下の手続きにより Power 法を用いて、最終的に W_n で X を計算する。

A ={$n \times n$ 行列の入力データ}

I ={最短路代数の下での単位行列を入力}

$W_1 = I + A$　(最短路代数の下での行列和)

$W_2 = W_1 \times (I + A)$　(最短路代数の下での行列積)

$W_3 = W_2 \times (I + A)$

：

$W_n = W_{n-1} \times (I + A)$

この Power 法は、最短路代数の下では、$X = (I + A)^n = I + A + A^2 + \ldots + A^n$ が成立することを利用している。

４．Sheet 上プログラミングの特徴

(4-1)　一連の求解プロセスを Excel Sheet 上に書き下すことにより計算処理を行うため、処理過程が視覚的に明示される。従って、ユーザにとって了解性にすぐれる。

(4-2)　ユーザからユーザへのソフトの継承性が高い。プログラムは説明書が存在しても、作成者以外には詳細は不明であり、プログラムの継承、改造は、従来から困難であった。

しかし、Sheet 上のプログラミングでは了解性の高さゆえにその問題はなくなる。

(4-3)　以上の(4-1)と(4-2)の特徴により、Sheet 上プログラミングは、PC を用いた授業でのデモならびに演習に最適である。すなわち、ブラックボックス化したソフトでは、入出力因果関係が不明であり、本来教えるべき内容をブラックボックス化しない Sheet 上プログラミングならではの特徴である。これは、**ヤッコー現象**（注１）、**GIGO 現象**（注２）の克服のために有効である。もちろん、PC を用いた授業以外にも、卒業研究、アルゴリズム開発、その他、システム開発、科学技術計算などの様々な場面で有効であることは言うまでもない。

(4-4)　Sheet 画面上の手続き配置を工夫することにより、また、デザインを工夫することにより、入出力が Sheet 上で一体となった、ユーザフレンドリな求解ソフトを設計できる。

(4-5)　複雑な機能、例えば、GOTO 命令文、DO-LOOP 等のセルにおける代入文は、現時点では無理である。又、ベクトルや行列の次元数はあらかじめ定めた場合しか対処できない。すなわち、次元数=*n* を入力する形式での処理は難しい。何が Sheet 上プログラミングで可能で、何が不可能かを整理する必要がある。

５．おわりに

アドインソフトのもつ複雑な処理機能ならびに内蔵プログラム機能は犠牲にして、それらの機能をできるだけ計算手続きの工夫により補完する求解アプローチとして、Sheet 上プログラミングを提案した。これは、丁度、サブルーチンの呼び出しを使わず、関数機能の組み合わせのみで処理実現を目指すものと考えられる（注３）。アドインソフトで可能な機能がこれによりすべて実現できるわけではなく、Sheet 上プログラミングには処理機能上の限界が存在する。しかし、将来の Excel 自体の機能向上ならびに我々ユーザによる計算手続きの工夫により、この限界が無くなっていくことを願う。

PCを用いる授業においても、Sheet上プログラミングは、単なるソフトのデモよりは有効であろうことを主張した。　OHPシートあるいはプロジェクタを用いる方法と比較して、思考プロセスを板書きで教授する方が、教育効果が高いのと同じであろう。すなわち、Sheet上プログラミングでは、アルゴリズム等の処理プロセス、思考プロセスがSheet上に表示されており、**考えることや「理論」の大切さ**を授業を通して実感できる。

最後に一言：数理的思考法の一つに、なるべく暗記事項をなくし、少ない基本原理から、様々な現象を説明しようという立場がある。提案する Sheet 上プログラミングは、思考プロセスの一部をブラックボックス隠蔽化せずに、すべて Sheet 上にインライン展開することにより、多少煩雑にはなるものの、明示的了解性の向上を図る一つのビジネスモデリング・アプローチと位置付けられる。

（注１）　ヤッコー現象〔2〕

計算機デモならびにコンピュータシミュレーションを用いた研究全般に対してよく言われるのは,「計算機を使って一生懸命やりましたけど,とにかく**ヤッ**てみたら**コー**なりましたが、何故そうなったか、その結果の妥当性は知りません」という状況に対する批判である.

ヤッコー現象と言われないためには，計算機デモを行う際に,計算プロセスを理解できる形で説明できること、因果関係の明示、理論と現実の双方への関連付けを意識すること、などが大事であると言われている.

（注２）　GIGO 現象

コンピュータを用いた仕事や研究の世界では

"Garbage in, garbage out" **「ごみを入れれば、ごみしか出てこない」**という戒めの言葉がある。つまり、計算機は与えられたデータを与えられた方法によって処理してその結果を出力するだけであり、データ内容や処理方法には関与しない。データや処理方法が不適切ならば、計算機は無意味・不適切な結果を**いかにも整った形式で**出力する。従って、計算機を用いて仕事をする際にＧＩＧＯ現象を回避するには、データ入力から、処理内容、出力解釈まで首尾一貫して、アルゴリズム担当者が責任を持って面倒みるべし…と言われている。『**ヤッ**てみたら**コー**なりました』の**ヤッコー現象**も、『ごみを入れれば、ごみしか出てこない』の**GIGO現象**、両方とも、計算処理の高度化・複雑化によりその中身がブラックボックス化して、入出力間の因果関係が判り難くなっている現状に起因する計算機技術者が陥りやすい過ちである。

（注３）　関数とサブルーチン

両者とも、同一反復処理を部品化して作るための仕組みであり、プログラムが複雑になってくる場合は、同じ動作の手続きを部品化することによりプログラムが簡素になる。同じ処理を何度も行わなければならない場合は、その処理を部品化しておけば手間が省けるし、ソースファイルがスッキリして見やすくなる。プログラミング言語によりニュアンス上多少の差があり、一概に言えないが、関数とサブルーチンの相違は以下の通りである。

１．　関数は単一の値や配列を返すが，サブルーチンはそれ自体値を返さない。
２．　サブプログラム内で計算結果の値を複数引き渡すことが可能なのがサブルーチンであり、ひとつしか値が得られないのが関数である。
３．　処理だけを記述し、戻り値を帰さないのがサブルーチン、処理後の結果を帰すのが関数である。
４．　サブルーチンは、プログラムの内部をモジュール化（部品化）する手法である。自分で関数を作成・定義するための手法ともみなせるため、サブルーチンは「ユーザ関数」とも呼ばれる。
５．　関数はコンパイル時に呼び出される位置に埋め込まれるように展開される。したがって通常のサブルーチンコールにともなうオーバーヘッドが少なくて済む。
６．　サブルーチンの呼び出しには、余分な手間、呼び出しのオーバーヘッド、がかかる。関数の狙いは、その手間を省いて処理を高速化することです。しかし、呼び出された先毎にコード内容が展開されるので、コードサイズは全体として普通に呼び出すより大きくなる。従って、関数とするためには、コードは短く、基本的な作業を行うものであり、システムコールなどの外部呼び出しにつながる部分は含まず、かつ頻繁に繰り返し呼び出される可能性が高い処理とすべきである。

参考文献

〔1〕川口博子、篠原正明、安藤直人：多変数 Goal Seek 機能のアドインソフトのインプレメント、日本オペレーションズリサーチ学会 2005 年度秋季研究論文集、２－B－２、pp.162－163 (2005.9)
〔2〕和泉潔 ：人工市場の作り方:ヤッコーにならないために, システム/制御/情報(システム制御情報学会誌), Vol.46 ,No.9, pp.547-554(2002).