講　　演

雑談的情報科学論*

清　野　武**

1. はじめに

いまから3年ほど前，私がこの学会の会長に就任したとき，慣例に従って，学会誌の巻頭言というものを書くはめにおちいったのであるが，そこでは“分極”という問題を取り上げてみた[1]．それらは(1)計算機の超大型化と超小型化，(2)情報科学の研究における虚学と実学，(3)情報科学教育における一般教育と専門教育，(4)学問体系における computer science と user sciences，(5)情報科学の社会的影響としてのコンピュートピア的楽観主義と情報公害的厭世主義，などで，一顧の価値もない雑談的議論にすぎないものであったが，それを読んで下さった会員のおられたことを知って，このようなものをあだやおろそかには書くべきでないことを認識させられたのである．

しかしそれにも増して驚くべきことには，なにげなく書いた言葉を，改めて想い出させるような，いくつかの論説や講演などに出会ったことが，この3年間にしばしばあったのである．これは学問や技術が発展の段階にある場合には，さまざまな分極が起こるのがむしろ当然である，ということを示しているのかも知れない．対立する極性の間の相互刺激的，あるいは相互補完的な作用は，いわば活力の根源として，きわめて重要な役割を果たしていると思われるからである．

2. 化学における分析と合成

分極化を扱った論説の1例として，この学会とは直接には関係がないようであるが，まず化学についての藤原教授の論説[2]から話題を拾ってみよう．化学における研究は，物性の測定と物質の合成という2つの極性を持っていて，これが，いろいろな場面に出てくるということである．そのほかに，化学という学問の1つの特徴として，研究に必要な情報が大量であり，しかもそれらが広い地域にまたがっていることが挙げられている．これらのことは，化学の研究における計算機の導入に当たって考慮すべき要素であると思われる．すなわち，大量の情報を処理するには，それに適したシステムが必要であり，広域にまたがる知識の利用のためには，データバンクとともにデータ通信や計算機ネットワークが活用されなければならないであろう．しかもそこに分析と合成という2つの極性が存在するということは，システムの構成にも何らかの影響があるに違いない．

これは見方によっては，応用情報学的な態度が強く要請される1つの場面であるといえるであろう．これについては，一般的な問題として，のちにもう少し触れてみたいと思う．

3. 思弁的数学と算法的数学

つぎに，数値解析の著書でも有名なスイスの数学者 P. Henrici 教授の講演[3]を中心に，2，3の問題を考えてみよう．これは“数学の研究・教育に及ぼす computing の影響に関する会議”(1973) で述べられたものであるが，その中で，今日の数学の危機というものが，2つの相反する力の間の不均衡に基づくことが指摘されている．数学における分極としては，古くから，(1)純粋数学対応用数学，(2)抽象数学対具象数学，(3)理論指向型数学対問題指向型数学，などが挙げられているが，Henrici 教授は，今日いちばん問題になっている不平衡は，(4)思弁的(dialectic)な数学対算法的 (algorithmic) な数学，の間の分極に存するものであることを述べている．

これは，他のいくつかの分極とも関連してはいるが，それらと同じものではない．まず思弁的な数学は，厳密に論理的な科学であって，そこでは，命題は真か偽かのどちらかであり，ある性質を持つ対象は存在するか存在しないかのいずれかである．これに対し，算法的な数学は，問題を解くための道具であり，そこでは，解の存在の証明だけでなく，実際に解が引き出されなければならない．しいていえば，思弁的数学は虚

* 情報処理学会第12回通常総会特別講演（昭和49年5月16日）
** 京都大学工学部情報工学教室

学的であり，算法的数学は実学的であるということになるであろう．これはまた，G. E. Forsythe 教授のいう数学の美しい面と役に立つ面に対応しているように思われる[4]．

もし思弁的に厳密な解のみにこだわるならば，月旅行の実現など望むべくもないわけで，このことはIFIP会長の Zemanek 教授が，“情報科学は技術の科学であり，真の技術とは妥協の技法である”[5] と述べていることと相通ずるものがあると思われる．

要するに，“思弁的な数学は人を冥想にさそい，算法的な数学は人に行動をうながす”ものであり，“思弁的な数学は洞察を生み，算法的な数学は結果を求める”という Henrici 教授の言葉は，情報科学にも当てはまるのではないであろうか．

世の人々は，技術あるいは工学に対して，ある程度の敬意を払うけれども，アルゴリズミックな行動よりも，哲学的冥想のほうがより高尚であると考える場合が多いようである．そして工学者自身，また技術者自身も，ときには抽象の世界に，あるいは冥想の深淵にあこがれることも否定できないであろう．このことはハーバード大学の A. G. Oettinger 教授が，“計算機科学は純粋数学よりもさらに純粋な要素から，エンジニアリングの中でも最もきたない要素までを含むものであり，それを恥とすべきではないし，逆に何でもサイエンスとよびたがるのは，1種の俗物根性である”[6] と警告していることを思い出させるのである．

4. 数値解析学の位置づけ

いろいろな計算に対して適切なアルゴリズムを作り出すことは，数値解析学の中心的課題であるが，一般の数学者は，めったにこの方面の名人にはなれない，という意味のことを Forsythe 教授は述べている[4]．これはまた，Henrici 教授がその講演で触れているように，数学におけるアルゴリズミックな才能は，決してダイアレクティックな才能のサブタレントではなく，教育においても，前者は後者とは独立に育成され，後者にプラスされるべきものである，という見解と一致している．

いずれにしても，数値解析学という学問，ないしは数値計算法という技術は，そもそも計算機が世に出た頃には，その第1義的な応用分野と考えられていたのであるが，Zemanek 教授は，“計算機が行列や微分方程式の処理ばかりに追われていた時代は，今や決定的に過去のものとなった”という意味のことを述べている[5]．1昨年開催された第1回日米コンピュータ会議において，数値解析の部門が設けられなかったことも，あるいはこのような考えの1つの現われであったかも知れない．

しかしこれらのことは，数値解析がいらなくなったとか，その重要性が薄れたとかいうことを意味するものではなく，数値解析という学問が，もはや情報科学の中心的課題ではなくなったこと，そしてむしろ，数値解析は，情報科学から独立すべきものであることを示唆しているように思われる．

これに関連して，わが国の高等専門学校における，いわゆる情報処理教育が，“FORTRAN+数値解析”だけに終わっている場合が多いことは，何とも了解に苦しむところである．専門学校に限らず，いわゆる情報処理教育の1つのねらいは，“情報化時代を主体的に生きる”ための教育であったはずであって，この目標が FORTRAN 教育だけで達成されるとは，どうしても考えられないのである．

蛇足ながら，この“情報化”という言葉は，日本語だけにあって英語にはそれに対応する単語がないといわれていた．この学会の大会でのパネル討論会でも，そんな話が出たことがあったと記憶している．ところが，昨年の夏，アメリカ人の書いた“日本の情報化”[7] という報告がドイツの雑誌に載っているのをみて，ついに情報化という日本語から“informationalization”という英語が作られたことを知って驚いたのである．情報科学とは直接の関係はないかも知れないが，ことのついでに申し添えておく次第である．

5. プログラミングについて

計算機の科学を特徴づける中心的要素として，プログラミングがあることは，昔も今も変わらないのであるが，これについての最近の話題としては，Dijkstra 教授の“structured programming”[8] を見逃すことはできないであろう．この有名な著書の中で Dijkstra 教授は，ある問題の解決に当たって，それを能率的に解くことのためだけに，知的エネルギーを浪費すべきでない，という点を強調している．たとえば，解の持つべき性質の完全なリストを作り上げようとすることは，しばしば知的浪費につながるものであるから，出発に際してはむしろ，解の持つべき性質のうちの極めて明白なものに着目するように注意をうながしているが，これはまさに，思弁的な立場と算法的な立場を区別する典型的な例であると考えられる．

周知の通り，この本の中には“8人の女王の問題”という面白い例題が載っているが，Dijkstra 教授がこれをある数学者に示したときの反応が興味深く述べられている．この数学者は，8×8 のチェス盤を 4×4 の正方形に4分割したとき，これら4つの正方形は，それぞれ2人の女王を含むべきである，という定理の証明にとりかかったというのである．

もしこのような命題が成立するとしたら，8人の女王の問題は，たいへんエレガントに解けるかも知れないという希望が持てるのであるが，しかしついにその証明は成功しなかったはずである．というのは，8人の女王の問題の解は 92 種類あって，そのうちには，4分割した正方形の中に，女王が2人ずついるような解もたしかに含まれているが，小さい正方形の中に女王が3人いたり，1人しかいなかったりするような解も，実はたくさん存在するからである．

6. 分極を結ぶ連続体

まえにも述べたように，分極という現象は，科学における共通的な傾向であって，それ自体が悪であるとはいえないのであるが，それが孤立化，あるいは compartmentalization という方向に進むと，しばしば困った状態になるのである．

情報の学問においても，“厳密性”ないしは“純粋性”を志向する動きと，“技術的”ないしは“実利的”方向への推進とがみられるように思われる．すなわち“純粋数学的情報学”と“工学的情報学”との間の分極が存在し，その間の橋渡しは，一般には，極めて乏しいということができるであろう．

Weingarten 教授[9]も指摘している通り，学問が成熟するためには，このような両極を結ぶ“連続体”すなわち橋が存在しなければならない．情報科学におけるこの種の連続体としては，おそらく ALGOL 60 以降に発展した計算機言語の設計，コンパイラを含む言語翻訳，言語理論などにまたがる1種の連続体を挙げることができると思われる．そしてこのような連続体が，いろいろな極性の間に形成されてゆくことによって，情報科学は真の成熟に向かうものと期待されるのである．

7. 応用情報学の立場

情報科学における別のレベルの極性として，“応用情報学”について考えてみたいと思う．ここにいう応用とは，“純粋数学”対“応用数学”という対立的概念における“応用”(applied) ではなく，“道具寄りの情報科学”(geräteorientierte Informatik) に対立する“応用指向的情報科学”(anwendungs-orientierte Informatik)[10] における“応用指向”(application-oriented) を意味するものであって，ケルン大学計算センターの Schmitz 教授の提案になる言葉のようである．

今日の計算機システムは，計算機という道具と，この道具を使って問題を解決しようとする利用者の間のコミュニケーションの手段として，プログラミング言語やオペレーティングシステムなどを持っている．そして従来は，道具であるハードウェアを所与のもの (Gegebenheit) として，これにいわゆる汎用 OS をつなぎ，本来の目的たる問題，すなわち現実のアプリケーションを，道具とコミュニケーションの手段のもとに従嘱させてきた，という傾向があった．これがすなわち道具寄りの情報科学の立場である．

しかし Zemanek 教授[5]も指摘している通り，“この世界がプログラミングシステムを指向するのではなく，プログラミングシステムがこの世に奉仕しなければならない”ことは明らかであり，この考え方は，ハードウェアを含む計算機システムにも当てはまるものである．これが“応用指向的情報科学”すなわち，ここでいう“応用情報学”の立場である．そしてこのことは，単に計算システムだけでなく，今後ますます重要になると思われる計算機ネットワーク，あるいは情報システムにも，そのまま適用されるはずである．

このように，従来ハードウェアないしは道具を Gegebenheit とみなす立場があまりに強すぎたという弊害があったからといって，道具寄りの立場を全面的に否定すべきでないことも当然である．すなわち，ここでも道具寄りの極性と，アプリケーション寄りの極性とは，相互補完的，相互刺激的に機能すべきであり，これによって好ましい効果が期待されるのである．

道具寄りの発想を支持する1つの根拠として，新しい道具なり手段なりを提供することによって，それなしには思いもよらなかったような応用や，新しい研究手法が開発されるかも知れない，という希望を挙げることができると思われる．つまり，これこれの目的に使うために，このような機械を作るのだ，というのは，いわば応用指向的な発想であり，何ができるか今はわからないけれども，何か予想外のことができるかも知れないから，これこれの能力のある機械を作ってみよう，というのが良い意味での道具寄りの発想とみることができるであろう．

後者は，どちらかといえばアメリカに多くみられる研究態度であって，ヨーロッパの学者が大いにうらやましがるところでもある[11]．これはもちろん，お金もかかるし，無駄も多いに違いないが，それなりの意義はあると思うのである．たとえば，TSSの初期に“失望落胆の1967年”[11]といわれた頃，多くの企業が相継いでTSSの注文を解約した中で，残ったのはアメリカのいくつかの大学だけであったということが伝えられているが，これが無駄であったかどうかは，歴史が告げてくれるであろう．

8. オペレーティングシステムについて

現在のいわゆる汎用オペレーティングシステムを応用情報学的立場から眺めると，あるいは若干の問題が出てくるのではないであろうか．

たとえば，OSのモジュラリティの必要性がさけばれてからすでに久しくなるのであるが，その効果は十分に発揮されているであろうか．モジュラリティのねらいとしては，モジュールごとのデバッギングやテストが容易で，システムの構成単位ごとの完成を速く，しかも確実にすること，また機能の追加や変更の要求に対して，柔軟に対応できること，などがあったはずである．これはまさに，今日いわれているプログラミングの構造化と共通の目標，すなわちソフトウェアの生産性をめざしていたものに違いない．

ところが，個々の利用者の立場からの注文に対して，今日のいわゆる汎用OSは，必ずしも弾力的であるとはいえないように思われる．このことは，ソフトウェアの生産性が依然として低いという事実とも無関係ではないであろう．

1例として，大学の大型計算機センターの立場から考えてみると，センター側あるいは利用者側からの要求のうちで，比較的簡単であると思われる変更や拡張に対しても，ソフトウェアシステムは意外にリジッドであって，個々のセンターに特有の運営形態に適合させることは，必ずしも容易ではない，という現実を知らされることが少なくないのである．ソフトウェアがリジッドであるとは，何とも皮肉な話であるが，事実はそうであるといわざるをえない．

いままでに述べたことから，応用情報学は合目的性の追求に重点を置いた立場であると解釈することができるのであるが，これにかかわるもう1つの問題として，コンパイラの最適化 (optimization) について，これも雑談的に述べてみたい．

最適化については，いろいろのレベルのものが考えられるが，ここではFORTRANコンパイラにおける普通の意味での最適化に限定して考えることにする．これはすでに，FORTRANのごく初期の段階で取り入れられていた機能であって，現在でも当然のこととして受け入れられている．すなわち，メーカはこの機能を売り物にし，利用者もこれについてさまざまな注文をつけるというのが実情のようである．

最適化は要するに，ソースプログラムに含まれている非能率的な部分を整理して，オブジェクトプログラムの実行時間を短縮しようという，まことに御親切な機能である．そしてこれが十分に存在意義を持ち，現実に利益をえている人も少なくないという，素直な主張も聞かれるのである．

しかしながら，少しくひねくれた考え方をすれば，これほど利用者を愚弄した話はないのであって，“お前のプログラムは駄目だからおれが直してやる”という前提に立っているように思えるのであるがどうであろうか．

オブジェクトプログラムの実行時間を縮めることについては，誰しも異論のあるはずはない．しかし，ソースプログラムの論理を変えてまで，それをしようとするのは，もはや最適化の域を超えたものというべきではないであろうか．

プログラミングに当たって，ある程度はコンパイルのプロセスを予想しながら，能率の良いプログラムを書くように注意するのは当然であるとしても，コンパイラが行なう最適化の仕掛けまで考慮して，うまくオプティマイズしてもらうように，プログラムを書かなければならないとしたら，本末転倒もはなはだしいのであって，これこそ知的エネルギーの空費ではないであろうか。

これについては，北海道大学大型計算機センターの長田，相良両氏による克明な調査[12]があり，今日のコンパイラは，ずいぶん行き届いた最適化を行なっていることがわかるのであるが，同時に，そのための“副作用”というべきものも，決して見過ごしえない性質のものであることを知らされるのである．

このように，かなり行き過ぎた最適化も現実には存在するのであるが，どこまでを行き過ぎとみるかは，議論の別れるところであると思われるので，極めて単純な，そしてたぶん最も古くから行なわれてきたと想像される最適化，すなわち，算術式の中の共通項の計算の重複を避けるという機能に限って考えてみよう．

これについては誰も苦情をいわないようであるが，しかしこの種の最適化によって，果たしてどれだけの利益が期待できるであろうか．それはむしろプログラマの責任に委ねるべきであって，システムが余計な世話をやくべき問題ではないと考えるのは間違っているのであろうか．

このほかにも，単項オペレータとしての“+，−”の処理，de Morgan の定理による論理式の簡単化，混合演算におけるオペランドの並べ換え，その他いくつかの最適化が，コンパイラによって利用者に（強制的に）提供されているが，その多くは，前述の共通項の処理と同じ性格のものとみてよいであろう．

もちろんこれらの最適化は，場合によっては利用者に利益をもたらすものであるし，少なくとも無害である，という主張は利用者の側にもあるであろう．しかしながら，このような機能がシステムに含まれている以上，システムは絶えず最適化が必要であるかどうかのテストを行なうわけであって，そのための時間的，空間的（メモリスペース的）損失は，結局は利用者にはね返ってくるということも考えるべきではないであろうか．

このように考えること自体が，あまりにケチというべきなのであろうか[13]．コンパイラが発見してくれるような，見えすいた無駄を，事前に調べることなしに何でも計算機に頼るのが知的エネルギーの節約というものなのであろうか．

Dijkstra 教授のいう知的エネルギーの浪費とは，いわゆる最適化によって簡単に救われるような性質のものを指しているのではないと思うのであるが，どうであろうか．

9. おわりに

とはいえ，ここでいくら“最適化はやめるべきだ”とさけんでみたところで，急に事態が変わるとは思われない．それはあたかも，Dijkstra 教授が，“FORTRAN などというものが存在していたことをわれわれが忘れ去る日は，早ければ早いほどよいのです”，……“FORTRAN はわれわれの思考力を浪費します．それを使うことはあまりに危険であり，したがって高価につきます”[14]（木村教授の訳文による），と警告を発したからといって，政府が明日から FORTRAN の使用を禁止する気づかいがないのと——次元は違うけれども——似たような話である．

FORTRAN に対する Dijkstra 教授の痛烈な批判については，私自身もとまどっているのであるが，ただ聞き流してすまされる問題とは思えないのである．

ひところ世間では，情報公害という言葉が流行し，計算機あるいは情報処理技術を諸悪の根源として非難攻撃する風潮があったが，批判の対象となった事例の多くは，日本でも外国でも，情報科学とは無関係なことであったように思われる．いわゆる情報公害は，多くの場合，システムの設計や，運営上の欠陥，あるいは，設計と運営との間の理解の不足，要するに応用情報学的配慮の欠除に基づくものであって，これは技術の未熟として責められるべき性質のものであろう．

しかしながら，情報科学およびそれを取りまく関連諸科学の世界の中での公害は，当然，情報科学の側にも責任があるとみるべきであって，知的エネルギーの浪費を多くの人々に強いるような計算機言語は，1つの公害として責められて然るべきである．

いわゆる“FORTRAN 公害”[15]とは，それほど高尚な思想から発したものではなく，もっと卑近な雑談的なものであるが，それでも結局は，教育の問題に帰着するという点において，共通しているといえるかも知れないのである．

ともあれ現実には，FORTRAN 公害は，浪費的オペレーティングシステムとともに，ますます広がってゆくであろう，という予感を禁じえないのである．そしてわれわれは，その中で生きてゆかなければならない，という宿命を負わされているのではないであろうか．

雑談的情報科学論がついに雑談の域を超えて放談的となり，会員諸兄の貴重な時間を浪費する結果に至ったことを許していただきたい．

参考文献

1) 清野：情報処理，12, 6, 319 (1971. 6).
2) 藤原鎮男：化学用計算機の設計，高分子，23, 295-296 (1974. 3).
3) P. Henrici: Computational complex analysis, Conference on the influence of computing on mathematical research and education (Aug. 1973, Missoula, Montana).
4) G. E. Forsythe: Pitfalls in computation, Tech. Report, CS 147, Stanford Univ. (Jan. 1970).
5) H. Zemanek: Was ist Informatik? Elektronische Rechenanlagen, 13, 4, 157-161 (1971).
6) A. G. Oettinger: University Education in Computing Science (A. Finerman, editor), Academic Press (1968).
7) B. O. Szuprowicz: Informationalization of

Japan, Angewandte Informatik, 317–321 (Aug. 1973).
8) O.-J. Dahl, et al.: Structured Programming, 1–82, Academic Press (1972).
9) F. W. Weingarten: Translation of Computer Languages, Holden-Day (1973).
10) P. Schmitz: Zum Standpunkt einer anwendungs-orientierten Informatik, Angewandte Informatik, 3–6 (Jan. 1973).
11) W. Giloi: Das Zauberwort Time-Sharing, Computer Praxis, 1, 130–134 (Jul. 1968).
12) 長田博泰, 相良 砌: 大型計算機センター運用会議資料 (1974. 2, 仙台).
13) 清野: 日立評論, 54, 3 (1972. 3).
14) E. W. Dijkstra: The humble programmer, Comm. ACM, 15, 859–866 (Oct. 1972); 木村泉訳: bit, 5, 11, 1243–1254 (1973).
15) 清野: bit, 4, 5, 365 (1972).