計算の哲学的意味

齊藤了文

序

ここで問題にしたいのは科学技術計算と結びつくコンピュータの利用である。計算機の科学 computer science でなく計算の科学 computational science[1] に焦点を合わせる。コンピュータがどのように作られているかとかその起源は周辺的な問題である。[2] この小論の目的は、von Neumann 以来のコンピュータの使い方である科学技術計算がどのような意義と限界をもっているかを明らかにすることによって、哲学的な科学理解に寄与したいということである。科学技術計算ということで、特に数値シミュレーションを扱う。理論、実験に対してシミュレーションが科学の第三の方法だと言われているからだ。

物理学は実験物理学と理論物理学に大別される。前者は、実験、観測機器を使って新たな自然現象やその規則性を発見したり、理論が提示する予測を検証する。後者は、様々な数学を使って自然現象の数学的構造を解明しようとする。しかし、複雑な物理現象を理解するのに数学的方法を使うにしても、「紙と鉛筆」（つまり解析的方法）だけでは太刀打ちできないことも多い。流体力学における乱流、量子色力学で扱うクォーク間の相互作用、銀河やブラックホールの生成などがその典型である。「このような物理系に対し、高速計算機による数値解析を行ってそのダイナミッ

クスを分析することにより、基礎法則からいかにして複雑な現象が生起するかを理解し、またその上で新たな自然現象の予言を行おうとする分野が計算物理学である。」[3] その意味での計算がどのような知識であるかを明らかにしたい。

まず確認しておくべきことは、現実の科学的探求においては、アインシュタインの相対論に匹敵するような画期的な思考が常に行われているわけではないということだ。多くの分野では通常科学が行われているとみなせる。そして通常の科学的探求の論理的構造は、仮説やモデルを提案し、その中で検証できる部分を取り上げて検証する。その結果が悪ければ仮説を変えたりするという枠組みで捉えられている。この枠組みは粗いものではあっても確かに働いている。そしてさらにある程度確証された法則や実験方法は分野ごとに多数存在する。多くの「科学的真理」を用いつつ更に科学的探求が進められている。そのような状況下で、コンピュータを導入して科学的探求をするときに何が起こるかを見ていこうとする。もちろん、学術出版や学術交流に果たすインターネットの役割[4]なども近年の科学的探求においては重要であるが、ここでは論じない。

科学における様々な営為、つまり法則を発見し、検証し、予測するときにコンピュータは使われている。コンピュータは記憶容量が多く計算が速い。しかし、科学的探求にとって透明な補助器具としてコンピュータを理解して済むのだろうか。コンピュータを科学的探求に使うための必要性を確認し、それがどのような意味で限定された道具になっているかをここで明らかにしていきたい。つまりコンピュータに計算を任せるということが科学的探求の一部としてごく自明で退屈なことをしているだけであり、哲学的には論じるに値しないことなのだろうかということを考えたい。そしてそれを踏まえて最後に問題にしたいのはシミュレーションである。

コンピュータは単にチューリングマシンの具体化とのみ見なされるような透明な演繹機械ではない。この点をまず第一節で確認する。第二節では、デジタルコンピュータを使った数値計算が普通の数学の計算と異なる点を概観する。それらを踏まえた上で、第三節ではシミュレーションを使う科学探求のどこに興味深い点があるかを考察する。第四

節では、以上の論点をまとめて、そこに含まれている哲学的含意を二、三示唆する。結局、コンピュータ科学は単なる応用論理とは見なせず、コンピュータの計算は通常の数学の公理からはずれており、予測という科学的探求は法則の正当化の解明とは違った面白さを持っているといったことが示される。このような主張を、「論定」するというよりも、研究の現状から「確認」していきたい。

一　道具としてのコンピュータ

まず、一般的に（数値計算に限定されず）コンピュータを使うことについて考える。つまり、コンピュータサイエンスの学問性をどう考えるかを考察する。これはコンピュータで具体的に計算して正確な計算結果を得たと言われる場合に何が起こっているかという問題だ。更にいえば、コンピュータをチューリングマシンの具体化とみなして、論理学と数学基礎論だけをコンピュータ科学の学問的な本質と考えて、その他の寄与はたいしたことがないと結論できるかどうかがここでの問題である。(5)

コンピュータは汎用の機械である。つまりテレビや電話や電灯のように使い道が限られているものではない。その意味で、コンピュータは道具として面白い特徴を持っている。(6) また、ハードの相違にかかわらず、多数のコンピュータで同様の仕事ができる。例えば、Windows というＯＳは様々なコンピュータのハードウェアで同じ使い勝手を実現している。その意味でコンピュータの道具としての特徴は基本的にはソフトに由来していると考えられる。(7) 従って以下「工学的」と述べる場合もハードの物理的性質を特に問題にしているわけではない。

以上のことを確認した上で道具としてのコンピュータの学問性を考えていく。その手がかりとしてコンピュータサイエンスという学問分野を規定してきたカリキュラムを見ていく。ACM（Association for Computing Machinery）はこれまで『カリキュラム68』『カリキュラム78』『カリキュラム88－プロトタイプカリキュラム』『カリキュラム91』

と、三度にわたる（88の本格的カリキュラムが91にあたる）カリキュラムを提案した。（後ろにつく数字は、それぞれの提案年度を示している。）「ACM カリキュラムの発展は、そのままコンピュータサイエンスの学問体系の発展の歴史的記録となっている。[8]」

ここで見ていきたいのは、同じくコンピュータサイエンスを学問と見なす根拠を与えようとしているにもかかわらず、『カリキュラム68』から『カリキュラム91』へと移行するにつれて、どこに学問の本質を見ているかが変化してきているということである。それは、結局は大きな流れとして、数学的根拠づけから、工学的根拠づけへの移行としてとらえられるであろう。

『カリキュラム68』においては、「コンピュータサイエンスは数学的思考や手法に大きく依存しているので、コンピュータサイエンスの大学課程は数学にその基礎をおくべきである[9]」とされる。つまり、プログラミングに関わる教育を、その数学的論理的構造のみに注目して教えようとしている。『カリキュラム68』の提案は、（平成三年の情報処理学会の委員会での評価では、）「既存の学問分野に対してコンピュータサイエンスが独立した学問分野であり、これを専門に教育研究する学科を独立して作りそこで教える内容がどのようなものであるかを明らかにしたものである。それから二〇年たった現在の時点でこれを眺めれば、数学的な面が豊富なのに対して、計算機分野については不十分の感があるのはやむを得ない。[10]」と言われている。

そして、『カリキュラム68』から二〇年以上たってできた『カリキュラム91』では、「数学と科学の理解はコンピューティングを中心とする学生にとって重要である[11]」と一言述べられているに過ぎない。応用や問題解決に「科学」を学ぶことが役立つということと並べて、数学はコンピューティング（広い意味でのコンピュータ・サイエンスがこう呼ばれることもある[12]）のいくつかの基礎的な話題を習得するために本質的であると言われている。数学の役割は、『カリキュラム68』に比べて、（基礎的役割を果たすことはもちろんであるが）相対的に低くなってきている。

さて、『カリキュラム91』（『カリキュラム88』も含めて）において特徴的なことは、実験、実習について特に取り上げて論じていることである。それ以前のカリキュラムにおいては、実験や実習に必要な時間数の指摘はあったが、それの意義については特にふれていない。どのカリキュラムも「学問」としてのコンピューティングを目指していたのだが、実験や実習を含めて「学問」として位置づけようとしたのは、『カリキュラム88』（これは、『カリキュラム91』のプロトタイプである）に特徴的な提案である。

情報処理にかかわる学生実験については、大学教育で行われる化学実験や電子工学実験や医学部の解剖実習などの実験とは違った性質をもつと言われている[13]。つまり、普通言われる科学の実験においては、「既に知られたことの再現、過去の科学者が行った実験の再体験」[14]が行われている。つまり、忠実に指導書に従った実験が行われる。いわば碁や将棋において定石を学んだり、絵画において手本を模写するようなものである。これによって、代表的な測定器具の使用法を習得したり、データの整理法を学び、実験結果を報告書としてまとめることを通じて実験の論理を学ぶ。

このような、マニュアル指導型の実験をすることがなぜ重要なのであろうか。囲碁や絵画を考えても分かるように、具体的な例では非常に多様性が生じる。物理学の実験においても、同じ実験をしているはずなのに、得られたデータは非常に相違することがある。様々のミスやノイズが生じる可能性がある。ここに実験技術の習得ということの意義がある。つまり、実験手順を厳密に同じにするということが、実験をやるうえで非常に重要になる。その意味で「再現可能性」ということが典型的な科学の実験の客観性を保証することになる。この点を理解するためにも学生実験が行われる。

それに対して、情報分野での実験を考えると、様々な実験のうちでハードウェアの実験は、電子工学科の実験と共通するものが多い。これは上述の実験と同じタイプである。しかし、アーキテクチャ関係の実験、プログラムの演習は、それとは違う側面をもつ。「簡単な計算機を設計して、プロトボード（かロジックトレーナ）で試作し、動作を

確認せよ」というような課題が与えられる。その指導書には、その計算機をどう作ればよいかは書いてない。ここが、上にのべた実験と違ってくるところである。伝統的な実験では指導書には「どうするか」がことこまかく指定してある。そういう手順を理解し、覚えることも目的に含まれるからである。しかし、いま述べた情報関係での実験では、「何をすべきか」という仕様はあっても、どう作ればよいかは指定されていない。ここに大きな違いがあるといえよう。これはプログラミングやソフトウェアになるとより顕著になる。プログラミングの課題は、こういうプログラムを作成せよという、仕様しか与えられていないのが普通である。[15]」これに似たものは、建築での設計、都市計画、工業デザイン、機械設計、美術の課題制作、音楽の課題作曲のようなものがある。

つまり、実験とはいえ、個別的課題についての指導書があるわけでなく、新規性が求められ、解も無数にある。[16]その意味で科学実験とは違ったいわゆる「設計」を学生に試みにやらそうとしているのである。つまり、一つの尺度だけでは判断できない「総合的」な課題に関わっている。これは、さきほど述べた科学実験とは異なり、設計と基本的に類似するいわば総合的、構成的な実験だといえるだろう。プログラミングにおいては、「再現可能性」はある意味では自明なことになる。誰がプログラムしても、同じ書き方をすれば、同じ動作が保証される。この意味で、実験をやるときにノイズやエラーは基本的には考えられない。(それに対して、自然科学では自然環境そのものがエラーを生じる原因となる。そのため、実験の方法を詳細に教えておいても、実験結果がうまく得られるとは限らない。)情報処理実習では、要求を仕様にまとめあげ、それを具体的にプログラムに書くことが問題である。過去の手法を知ることは重要であっても、そこから先のことが実験には要求される。[17]その意味で「総合的な価値判断のできる能力を身につける」ことが、実習を通して要求されているのである。つまり、設計を一つの典型とする工学的な実験をプログラム演習ではやっているのである。

さて、『カリキュラム88』においては、コンピューティングを学問として成り立たせるための三つのパラダイムが

提案されている。

「第一のパラダイムである理論（theory）は、数学にルーツをもち、次の四段階を経て、首尾一貫した妥当な理論を展開する。

1　研究対象を特徴づける（定義）
2　それらの間での可能な関係を仮定する（定理）
3　関係が真理であるかどうかを決定する（証明）
4　結果を解釈する

たとえばエラーや矛盾が発見される場合に、数学者はこれらのステップを繰り返すだろう。

第二のパラダイムである抽象化（abstraction）は実験科学の方法にルーツをもち、次の四段階を経て、現象の解明に導くものである。

1　仮説を形成する
2　モデルを構成し、予測を立てる
3　実験を設計し、データを集める
4　結果を解析する

たとえばモデルの予測が実験結果と一致しない場合に、科学者はこれらのステップを繰り返すだろう。「モデリング modeling」とか「実験 experimentation」とか呼んだほうが適切である可能性もあるが、ここでは計算機分野ではふつうそういわれているという理由によって、このパラダイムを「抽象化」と呼んでおくことにする。

第三のパラダイムである設計（design）は、工学にルーツをもち、次の四段階を経て、与えられた問題を解くシステムないし装置の構築に導くものである。

1　要求を述べる
2　仕様を述べる
3　システムを設計しインプリメントする
4　システムをテストする

たとえば、テストの結果、システムの最新バージョンが要求を充分には満たしていないことが分かった場合には、エンジニアはこれらのステップを繰り返すだろう。」[18]

ここでは、三つのパラダイムがあるということが重要である。「理論は数理科学の基盤である。応用数学者は、健全な数学の基礎なくしては科学の進歩はありえないという見解を共有している。抽象化（モデリング）は自然科学の基盤である。科学者は、科学の進歩は主として仮説を立て、モデル化のプロセスを組織的に遂行し、それらを検証、評価することによって達成されるという見解を共有している。同様に設計は、工学の基盤である。エンジニアは、進歩が主として問題を設定し、設計のプロセスを組織的に遂行することによって、それらの問題を解決するシステムを作り上げることにあるという見解を共有している。」[19] そのそれぞれのパラダイムが、数学、科学、工学の基本的な方法論を示している。そして、コンピューティングでは、この三つのプロセスが密接に絡み合っており、そのためにどれか一つだけを基本的なものとして取り上げることはできない、とされる。[20]

抽象化と設計というパラダイムを取り上げたことが、実験や実習をカリキュラムの中に特に取り上げた理由になっているのだろう。そしてもちろん、ここで「工学」とはいっても、通常の物理法則にかかわるシリコンチップの物性などは問題にされない。あくまでも、情報の処理にかかわる限りでの工学が問題になっている。（先に、情報分野の実験は、科学の実験とは少し違うということも見てきた。）少なくとも、学問としてのコンピューティングには、数学、さらには数学基礎論の一部とみなすことですまない問題が含まれていることははっきり意識されてきたように思

われる。その点について更に「ソフトウェア工学」とダイクストラの論点を含めた「プログラムの検証理論」を取り上げて論じることにする。

大規模なプログラムを扱わねばならない状況において、数学的論理的学問性と違うものが取り上げられるようになった。「一九六〇年代に入って、多人数を要する大規模なソフトウェア開発が行われるようになってきたけれども、開発の当初に意図したようにはシステムが実現できず、その性能、信頼性、納期、費用などの点で多くの問題が顕在化しつつあった。」[21] こういう問題状況の中から、ソフトウェアの開発や保守に関する技術や管理の方法の考察が始まり、ソフトウェア工学が提唱されるようになった。「ソフトウェア工学は、ACM カリキュラム68が発表された当時には存在せず、ACM カリキュラム78が発表された後になって研究が進んだ。」[22] 問題は、一人で作ることのできない規模のソフトウェアの開発に関わる。大規模な複雑性の処理が問題である。まず、発注者が頭の中に描いているものと、受注者が発注者の要求として理解したものとが一致していなければならない。大規模なソフトで開発に携わる人が多い場合には、これは単純な問題ではない。数学や論理学においては、あまり問題にされなかったことが、大規模な複雑性に直面することによって、問題として取り上げられて、ソフトウェア工学が成立したのだ。[23]

大規模で複雑なコンピュータプログラミングにおいては、例えば保守の問題が生じてくる。それは、上流工程と下流工程の乖離がその原因となっている。上流工程つまり外部仕様記述の段階（例えば、顧客が時間割作成のプログラムを組んでくれと要求する）では顧客との意思疎通をはかるためにも図的な記述などが行われる。それを下流工程ではプログラミング言語（FORTRAN などに典型的な逐次処理言語）で処理できるように変換しなければならない。そしてこの変換がシステムプログラマの恣意に任されている（プログラマの技能に依存する）ために、プログラムの保守とプログラムの理解（プログラムからシステムの機能と振る舞いを理解すること）が困難になる。さらに、デバッギングや小修正がプログラミング水準で行われるために、実在するプログラムは内部仕様とさえも乖離の度を増し

つづける。こうして、本質的に副作用が避けられないために、部品化を困難にし検証と試験とを極度に困難にしてしまう。これが、ソフトウェアの生産性の向上を妨げ、ソフトウェア・システムの保守容易性の低下を招いている。[24]

また設計に関しては上に述べた上流工程つまり要求仕様を決定することそのことが非常に難しい。ある現象のモデル化、より工学的な言い方では要求仕様を決定することはなかなか難しいことである。（もちろんモデル化することの難しさはコンピュータと特に関わらなくても生じる話である。[25][26] ただコンピュータは汎用機械であるために、その仕様の決定が大きな意味をもってくる。）つまり、人間が実現して欲しい機能を明確にすること自体が非常に困難なことである。人工知能のエキスパートシステムを作ろうとしたときに、エキスパートがどのような仕方で推論しているかを聞き出すことが非常に困難であり、しかも意識化して言葉で表現してもらったものは、エキスパートが実際に行った推論過程とは違っていることが見出された。これとよく似たことがここでも起こっている。しかも、プログラムの「正しさ」を考えようとすると、スペルミスなどの単純なミスによってプログラムが走らない場合はともかくとして、（不明確な）顧客の要求にどれだけ応えたかが最大の問題になる。だからこそ、要求仕様の決定がプログラムの正しさの中心課題になるのだ。以上のように大規模プログラムの保守と要求仕様の決定がソフトウェア工学の守備範囲になる。

そして、大規模な複雑性に対処するための学問を作り上げることが、『カリキュラム91』を特徴づけるものだと思える。「計算機分野は、数学および工学に深く根を張っている。数学は解析を、工学は設計を、そこにもたらした。計算機分野はそれ自身の理論と実験方法と工学をもっている。この点計算機分野は、たいていの物理的科学において、科学とその発見を応用する工学が（たとえば化学と化学工学、というように）別物とされているのと対照を示している。この分野において科学と工学が不可分なのは、この分野の科学的および工学的パラダイムが、根本的な相互関係をもっていることによる。[27]」この意味でソフトウェアが扱おうとする問題は数学的学問性が問題になる以前の状況に

対処しようとしている。しかも、これは学問性にとって無視できる視点ではなく、実際にプログラム[28]を作ろうとする場合にはどうしても省略しえない視点である。従って、それも含めた上でコンピュータサイエンスの学問性は何かを考えねばならない。

さて、次にダイクストラの論点を取り上げる。一九八九年二月、ダイクストラ（Edsger Dijkstra）は、『計算科学を本当に教えることの残酷さについて』という講演を行い、ACMの『カリキュラム88』に対して問題を投げ掛けた[29]。これに対して何人かの同僚と討論することになった。ダイクストラは、数学的証明を中心にして、コンピュータサイエンスを理解しようとしている。しかし、それに対して何人かが反論している。この論点の理解は、『カリキュラム88』の論点を明確にするためにも重要である。

ダイクストラの問題状況の把握は、計算機は我々の歴史において過激な新規性（radical novelty）を表しているというものである。そこには二つの過激な新規性があり、一つはプログラムの部分となる一ビットと大規模なプログラムの数百メガバイトという比のとてつもない大きさである。これは飛行機の部品と全体との比に比べてはるかに大きい。「第二の過激な新規性は、コンピュータが我々には初めての大規模なディジタル装置であるということだ[30]。」通常の機械には（例えばボルトとナットの間に）遊びがあるのに対して、「ディジタル装置」は、一ビット変化しても（例えば十と一が変化すると）、劇的な結果を引き起こすことがある。このような理解に基づいてダイクストラは、計算科学を形式数学と応用論理の方向に位置づけるべきだと主張する。ただし、計算科学は、形式的方法の効果的使用に関心があり、しかも普通の数学の式よりもずっと大規模な式を扱うという点に注意しなければならない。ダイクストラは、計算科学をVLSAL（Very Large Scale Application of Logic: 大規模応用論理学）と見なそうとする[31]。基本的なアイディアは、大きな集合について何かを示そうとするときには、その要素を個別的に扱って、しらみつぶしの探索をするよりも、一般化された定理を証明する方がいいということである。「形式的な構文規則と意味論を定義する証

明規則をもったプログラミング言語は、形式的なシステムであり、それに対してプログラムの実行が一つのモデルを与えるだけである。[32]」こういう意味で、プログラムはプログラムを書くときに、形式的な証明を与えることも同時にやるべきである。これが、プログラミングコースが形式数学のコースの一つだというダイクストラの主張である。

この主張に対して数人の研究者がそれぞれ少し違った立場からダイクストラの考えにともに反対している。その中の二人をここでは取り上げる。エムデン（M. H. van Emden）は、「仮に我々が常にプログラムを形式的に規定してそれらが正しいことを証明する段階に到達したとしよう。その場合、我々はソフトウェアが抱える問題の小さな部分を消去したにすぎない。大きな問題は、我々が何を望んでいるか正確に言うのが難しいことである[33]」と述べ、ダイクストラの形式的方法の限界を指摘する。複雑なシステムにおいては、「我々が望んでいることを本当には知らない[34]」。それを発見する一つの方法は、プログラムを走らせることである。またハミング（R. W. Hamming）は、「ダイクストラが、プログラミングは数学に似ており、数学とはユークリッド流のものであると信じている[35]」ことを最大の問題だと考えている。「プログラムを実行する前に正しいことを「証明」するという彼の考えは、ペーパーマシン上のペーパープログラムには適用できても現実の機械にはできない[36]。」「工学においては、設計の提案と現状の実践で何ができるかの間には、「もちつもたれつ（give and take）」の関係がある[37]。」そのために、書かれたプログラムの正確な記述から始めることはできない。

このような彼らの問題点の指摘に対して、ダイクストラは、数学の将来像の可能性について、非常にオプティミスティクな返答をしている。しかし、私の見る所ダイクストラに対する反論の方がプログラミングの現状把握としてより実際的であるように思える。ハミングの論点の前提にあるのは、設計がアイディアの単純な実現とはいえないということだろう。つまり、現実に合わせて修正を加えることが、「設計」の中心課題だという考えである。これは、エムデンの「我々が何を望んでいるか正確に言うのが難しい」という論点と結びつき、モデルの変更、仕様の変更をど

のようにしてうまくやるかということが、形式的証明以上に大きな意味をもつのである。[38] この意味でプログラムの正しさを大規模な応用論理学という仕方で保証する試みは難しいように思える。

数学や論理の位置づけという基本的に同じ論点をめぐる論争が一九八九年以降行われていた。それは、プログラムの検証 Program Verification がコンピュータサイエンスにとってどういう意味を持つかという論争である。一方の論点は、C. A. Hoare のものである。「あるプログラムの特徴と、任意の与えられた環境の中でプログラムを実行した結果が全て、原理的に純粋に演繹的な推論によって、プログラムというテキストそのものから見出されるという意味で、コンピュータプログラミングは精密な科学 exact science である。[39]」論理学がコンピュータ科学を基礎付けるというこのホアーの考えに対立する考えが「コンピュータ科学は経験科学である[40]」というものである。さて、もともとこの論争が生じたのは J. H. Fetzer という哲学者が、プログラムの検証は確実性を与えるものでないから不必要であり、場合によっては有害だという内容の論文[41]を一九八八年に書いたことに由来する。それに対して否定的な反論がその論文を掲載した CACM という雑誌に多数寄せられて論争が生じた。フェッチャー自身は右の内容そのものが彼の意図をうまく表現していないという論点も含めて多数のコンピュータサイエンティストと渡り合っている。しかし、最終的に決着した論点を見る限りは、コンピュータ科学にとってのプログラムの検証理論は、数学にとっての証明の役割を果たすことはできないという共通理解は得られている。せいぜい、自然科学にとっての数学の位置づけと同じようなものとされている。[42]

一般に、ソフトウェアをどう位置づけるか。実験をしないとすると、どのような正当化をしているか。再現可能性はあるといえる。それでは学問として何が問題なのか。このようなことを問題にしなければならない。プログラムの検証という仕方で、数学の証明を目指してソフトウェアの質を保証しようという試みもある。しかし、この試みは大規模なプログラムではうまくいかない。[43] 論理的な保証があれば数学的な学問性があるとみなしうる。しかし、大規模

なソフトウェアの場合には他の種類の保証が必要になる。プログラムの検証の理論も、プログラムの規模が大きくなるとほとんど無力であると言われている。[44] そのため、検証とテストも、大規模なプログラムの場合には難しい。このような大規模なソフトウェアを実際に扱わねばならない。そのためコンピュータサイエンスはその学問性を論理にのみ求めることはできなくなってきたのである。

結局、数学や論理というよりも、工学という特徴づけが重要だ。カリキュラムの変遷と、新しい学問として必要とされるようになったソフトウェア工学の現状を見るとき、そしてプログラムの検証理論の位置づけを考慮すると、事実認識として、コンピュータプログラムの正しさを保証する基礎を数学や論理学にだけ見ることは問題を含むことが理解される。チューリングマシンの具体的実現として理解するには、コンピュータのプログラムは余りにも複雑になっている。そのために、コンピュータの働きを理解しようとするときも、数学的な一般論の一つの具体化とは見なせないことが生じている。コンピュータをチューリングマシンの応用と見なすにはそこから排除される部分が余りにも大きくなっている。コンピュータシステムの故障や暴走も、この論点抜きには理解できないだろう。

二　数学と数値計算

完璧な宇宙方程式が与えられてもそれを解けないならば何になるだろうか。ニュートンの運動方程式を解くことによって落体の法則を距離を時間の関数として表現することは、高校の物理で理解できることである。しかし、このように微分方程式が与えられてもそれを解くことが非常に難しい場合がある。それは一般に非線形微分方程式と言われるものである。その典型が後に述べる天気予報に出てくる流体力学の方程式である。このような方程式は、一般に解析的に解が求まることはない。そしてよく知られた円や楕円のような方程式で解が表現されていないならば、たとえ解析解が求まったところで全体の軌跡をたどることは容易ではない。そして多くの場合には解析解が見つからないた

めに数値解を求めることで満足しようとする。これらのことは、微分方程式（があってもそれ）を解くことによって未来を予測することが現実には非常に困難であることを意味している。

もう少し具体的に、微分方程式を考えよう。物理学の法則は基本的に微分方程式で表わされるものが多い。そして数学において研究されてきた解析的な理論は微分方程式の解に関する多くの知見を与えてくれる。例えば、微分方程式を解くことによって位置が時間の関数として求まったら、そこに時間の具体的な値を入れると、その時そのものがどこにあるかが分かる。これが法則を使った予測のイメージである。しかし、具体的な微分方程式が与えられたときに、解析的な解の表現が困難であったり、初等関数や特殊関数の組み合わせでは解が表現できないことがある。またたとえ、解が解析的に求まったにしても、解が陰関数として求まったり、複雑な形をした関数で表わされていたとすると、その解の振舞いを想像することは困難である。どのような法則が存在するかを問題にするというよりも、法則を用いて自然現象を予測しようとするとこの点が問題になる。このような場合には微分方程式の数値解法が一つの手がかりとなる。「フォン・ノイマンが数値解析にきわめて大きな関心を抱いた理由の一つは、それ以外の方法では実質上、流体力学を研究することができないということであった。流体の流れを支配する偏微分方程式は解析学の手に負えないし、実験的研究するのも往々にして実際的でない。それゆえ、原子爆弾の設計、航空工学、天気予報その他の、流れに関する問題にとっては、数値的アプローチが唯一の頼みの綱である、ということが多い。」(45)

数値計算においてもまず注意すべきことはどんなに速いコンピュータでも計算の速さや記憶量には限界があるということである。数学者や論理学者は昔、チューリング機械による計算可能性の理論を与えた。しかし、それは実際的な問題を扱う技術者を満足させるものではなかった。というのは、理論的には計算可能とされる問題のうちのかなりのものが、その計算量が膨大であるために、現実の計算機では計算できないということを技術者は経験していたからである。そこから計算に要する時間量や記憶量に基づく計算の複雑さという新しい理論が数学の内部に出来てきた。

この理論によれば、入力情報の長さの多項式オーダーの時間量やテープ量では計算できないと考えられる問題が数多く存在することが明らかになった。例えば、指数関数のオーダーの時間量が必要となる計算は入力情報が少し大きくなるとどんなに速い計算機でも手に負えなくなる。スケジューリング問題や割り当て問題の多くはこの仲間に入る。[46] 現実のコンピュータを使う限りはこの制約を忘れることはできない。これはコンピュータ処理の有限性の問題である。

ある瞬間にどういう力が働いているかとかどのような場（電場、磁場など）の中にいるかということについては我我は予め多くの物理学上の知見の蓄積があり、更に計測によっても多くの情報が得られることが多い。そのため、それらに関わる微分方程式を立てることはこれまでの蓄積とデータさえあれば比較的困難なことではない。しかし、その意味で数理モデルを立てることは容易になってきても、その微分方程式を解くことは困難な場合が多い。特に、非線形微分方程式は一般に解を求める方法がなく、特殊な問題でなければ解析解（多項式や三角関数などの良く知られた関数を使って表現できる解）は存在しない。つまり、解に数値を代入して計算すれば答えが求まり、本来の位置が計算でき予測ができる、というような単純な状況にはない。にもかかわらず解を求める必要がある。そのものにどのような力が加わっているかが分かっただけでは、そのものの軌道を予測することはできないからである。従ってそこで使われる方法が数値積分である。微分することによって接線の傾きが分かるのだから、最初の位置が決まればそこから接線方向に「ほんの少しだけ」進んだらそれは求めたい軌道と「そんなに」違わないであろう。だから、その進めた位置を今度は出発点として同じことを繰り返せば実は求めたい軌道（に近いもの）が得られるはずだというのが数値積分の基本的なアイディアである。接線をたどっていけば、少しはゴツゴツしたグラフであっても、求めたいもともとの軌道に近いものが得られるはずである。[47] そして軌道が得られるということは、そのものの位置が分かるということであり、位置の予測が可能になったということである。（ちなみに接線をつなぎ合わせてもとのグラフを再構成する試みが数値積分と呼ばれるのは、微分と積分が逆演算であることを思い出せば理解できる。もとのグラフを微

分して接線の傾きを求めたのである。）このように次々と求める間隔を小さくして同様の計算を繰り返すことはコンピュータの得意とするところだからコンピュータの発達によって数値積分が可能になり、非線形微分方程式の特性も様々に分かるようになってきた。もちろん数値積分は具体的な数値を出すのであるから、通常の理論のように一般論がすぐにできるわけではない。そしてもちろん、個々の数値積分でも手計算では到底無理な繰り返し計算が必要となるので、計算量の制約が重要となる。しかも、一般的な傾向を見ようとすると、具体的な数値積分を何度も繰り返して、多数のグラフを生成する必要がある。

もう一つの問題は、デジタルコンピュータの本性としての有限性であり、その「離散的」性質だ。これはコンピュータの内部表現において一般的に実数を表現して計算することはできないということだ。コンピュータ演算の特徴は次のようにまとめられる。[48]

① 本質的に有限桁の数表現のみが可能で、そうした数を有限個記憶することができる。

② これらの数体系の上で、四則演算（算術演算）と論理演算のみを実行することができる。

③ 演算の処理速度は非常に高速であり、かつ記憶できるデータの量は有限といっても非常に大量であって、こうした点では人間の能力をはるかに凌駕する。

例えばコンピュータによって行われる基本演算は、一定の桁数で丸められる（例えば四捨五入、切り捨てなど）ために、四則演算に関する結合法則や分配法則は必ずしも成り立たない。[49] 例えば、$2\times(1/3)-(2/3)=0$が成り立たない。コンピュータでは数を有限の桁で表わすためにその最後の桁が存在し、それは$1/3$では3でありその2倍は6である。それに対して、$2/3$の最後の桁の数字は6でなく四捨五入のために7になっている。そのため差し引きしても0にはならない。だから、その意味では厳密に言えば、初等数学の成り立たない世界をコンピュータは扱っている。自然数論（本来は有理数論、実数論）の公理を満たさない計算をしている。公理を満たすような対象を数と考えると

すると、コンピュータの扱う数は本来の意味では数といえないことになる。[50]

もちろんコンピュータでは近似値のみを扱っていると考えれば有効数字を二〇桁もとって、その最後の数字が1ぐらい違ってもたいしたことはないと思うかもしれない。手計算の場合には数値的安定性が深刻な問題になることはまずない。実行できる計算回数に限界があるからだ。[51] しかし、コンピュータを使うことによって、何百、何千万回もの乗算が瞬時に行えるようになった。これが丸め誤差の累積を生じる可能性をもたらしたのだ。コンピュータの能力が問題を生じたのだ。このようなことの生じる問題は「不適切問題 ill-posed problem」と呼ばれる。(第三節では計算安定性に関する問題として取り上げる。)

また、通常の数学で用いられる極限操作、収束、無限小、無限大の概念が、解析学とは異なった意味で、厳しい評価を伴って用いられる。[52] 解析学の中間値の定理のように、解の探索でこの範囲にあるということを示すアルゴリズムは中間値の定理を利用している。連続なモデルでは収束が使える。しかし、コンピュータのアルゴリズムでは、有限の操作をしているためにそんなにうまくはいかない。計算モデルを作ろうとする場合には、このような問題にも注意しなければならない。

「いちどある特定の種類の機械を使って、数値計算で問題を取り扱うことが決まれば、すべての超越的な演算は、基本的な演算——機械にとって基本的な、すなわち計算機で直接取り扱えるような演算——を使って書き直さなければならないし、すべての極限操作は適当なところで打ち切って、基本演算の有限列に置き換えなければならない。[53]」これは物理学を含めて一般に科学の記述方法としての微分方程式が、そのままの形では解を求められないことを意味している。

従って、微分方程式の数値解法が必要になる。しかし、解法として注意しなければならないことは、優れたプログラムに要求される次のような条件を満たすことは、相互に矛盾した要求でもあり、難しいということだ。[54]

① できるだけ一般的な常微分方程式と初期条件に適用可能であること。
② 局所的にも大域的にも高次の近似解を与えるアルゴリズムであること。
③ 局所的にも大域的にも数値的に安定なアルゴリズムであること。
④ できるだけ少ない手間を費やすことで、〝速い〟アルゴリズムであること。
⑤ 分かりやすいアルゴリズムで、使い方もやさしいこと。

つまり、科学的に方程式を確定することができた後で、それを解く段になると以上のようなトレードオフを考慮しなければならなくなる。（ここに数学以外の基準が入ってくる。）実際に方程式が見つかってもそれを使う場合には、厳密性が主張できなくなることもある。これは、計算を「しっかり」行うことが難しいことを意味している。つまり、法則が与えられた上での予測やコントロールに関する事柄は、以上のような論点を見るだけでも法則の確定とは違った難しさを含んでいる。

こうして、計算処理時間の有限性の問題と有限の表現を利用するという問題の二つが組み合わされてコンピュータの数値計算の難しさと特異性が生じている。(55)

また、市販されている精緻な解析用計算機プログラムも、それ自身大規模なプログラムである。そのために第一節で述べたようなプログラムの正当性の評価という問題が残ることになる。解析担当者にとってはプログラムの内容はブラックボックスのままで、取扱説明書にもとづいて使用することになる。その場合、対象とする解析への使用の適否、バグの有無についての検討は実際上不可能に近いことになってしまう。

今述べたコンピュータの具体的な使い方はこれ以上問わないにしても、コンピュータという機械を使って解析を行おうとする限り、コンピュータが具体的に含む様々な制約を考慮する必要がある。(56)「有限」の方法で数学の無限の問題を扱おうとするときに問題が生じる。デジタルコンピュータの特徴である離散性と計算量の限界という具体的にコ

ンピュータを扱うことに由来する問題は基本的にコンピュータの有限性に由来する問題である。さらに、特に非線形微分方程式を解くときに生じる数値的な安定性の問題は、予測し解を求めるときに生じる問題である。微分方程式が与えられれば「原理的に」世界の全てが分かったという仕方で満足する人にとっては有限性や安定性の問題は些細な論点かもしれないが、この世界に対処しこの世界に生きていこうとする人にとっては、解が具体的に必要であるために、有限性の問題は大きな論点になる。

三　数値シミュレーション

第一節と第二節では、コンピュータを使うこととその場合の数値計算の問題点を見てきた。もちろんコンピュータを使わなければこのような問題点は生じないといわれるかもしれない。しかし、実際にコンピュータを使わなければ扱えないような「計算」は科学技術の多くの分野で非常に多い。その使い方の一つの実例が、数値シミュレーションである。これは「物理対象そのものではなく、それを表わす数学的モデルを扱うという点では理論物理学と共通点をもつが、データを取得し、分析し、演繹的にではなく帰納的に推論するところが実験物理学と共通している。計算物理学が「第三の物理学」と言われるゆえんである。」[57]

ここでは具体例として、電子計算機誕生のころから取り上げられている天気予報に関する数値シミュレーションを例示することにする。

気象学の歴史から言うと、ノルウェー学派（ベルゲン学派）によって温帯低気圧モデルが提唱された一九二三年以降、数値予報が実施されるようになってきた一九六〇年代までの時代は、気象学では総観気象学の時代と呼ばれている。彼らは、地上の天気図に描かれた高気圧と低気圧の東進を総観的に追跡することによって天気予報を行おうとした。実は、このような提唱が起こる以前の一九一〇年代にライプチッヒ大学で流体力学の方程式を実際の大気の流れ

にも適用して、予測を行う気象学を確立することによって、気象学を精密科学にしようとする試みが（主流ではなかったが）存在した。その流れを汲んだのがイギリスの Richardson であった。彼は一九一〇年代の考えをまとめて一九二二年に "Weather Prediction by Numerical Process" という本を出版し、大気の流れをどのような計算方法で予測するかを提案した。(58)

リチャードソンの数値天気予報をもう少し詳論しよう。(59) 天気予報をするのに酸素や窒素や水などの分子の動きを基にするのは到底無理である。コップ一杯の水でも一〇の二五乗個もの水分子を含んでいる。このそれぞれの粒子の運動を追跡することはおろか初期値を求めることも無理である。だから大気のマクロな相互作用を基にしてモデルを作り上げる。天気の動きを支配する低気圧や高気圧の発生や動きは、大気の流れと温度や気圧の変化に依存する。だから、それらを支配する流体力学や熱力学の方程式を時間積分することによって天気予報が可能となる。このようなアイディアは、二〇世紀初頭には分かっていた。リチャードソンはこのアイディアに従って、差分化した方程式をヨーロッパ大陸上に置いたメッシュを用いて解こうとした。これはコンピュータのない時代におけるシミュレーションの試みである。彼はヨーロッパ付近のごく限られた領域について計算しただけだが、六時間先までの予測計算に一ヵ月以上の時間を要した。しかし、時間のことはおくにしても彼の予測は失敗に終った。この理由を見ることによって、シミュレーションを行う場合に考慮すべき問題を取り上げてみる。

それは、「大気中に低気圧のような渦運動と水面の波に相当する運動が混在することに注意していなかったこと、計算安定性についての CFL 条件(60)が知られていなかったこと」(61)という二つの問題点に由来するとされる。つまり、観察などを含めて理論がある程度成熟していることと、数値解析に関する論点が解明されている必要があるというのだ。(62) リチャードソン自身は失敗の原因を初期条件の精度の悪さに求めていたがそれだけではなかった。従って、第二次大戦前後にラジオゾンデ観測網ができて、上層大気の流れや温度分布が常時観測されるようになることだけでは数値天

気予報の確立には十分ではなかった。さらに、大気運動の立体構造が解明され、温帯低気圧の発達を南北の温度差に起因する不安定性とする理論や、ロスビー波と呼ばれる大規模な渦度の波の理論が明らかになる必要があった。このように天気の力学についての理論が進展することによって、天気の移り変わりの変化を予測するためのモデルや近似方程式に何が必要かが分かってきた。こうしてようやく数値天気予報にまともに取りかかれる準備ができたことになる。つまり一九四〇年代の後半には低気圧の発達に関するメカニズムが、天気図解析からも理論面からも明らかになってきた。このようなことを踏まえた上で（準地衡風モデル[63]を用いて）、フォン・ノイマンなどはコンピュータを使った数値天気予報をしようとした。

温帯低気圧は、赤道と極との間の温度差によって生まれる。それは、熱を極に運んで温度差をなくすように機能する。さて、温帯低気圧の数値予報モデルが与えられると「その大気モデルに極と赤道の加熱差を導入して長時間積分を実行すれば、ひとりでに低気圧が生まれ、実際の大気と同じような流れのもようと温度分布とが得られるはずである。[64]」この大気大循環の数値モデルによるシミュレーションは一九五六年 N. A. Phillips によって初めて行われた。彼は大気を上下二層に分けたモデルを使って三〇日の積分を行って、「温帯低気圧をはじめ、中緯度上空の強い西風ジェットストリームや低緯度の北東貿易風など、大気大循環の基本的様相がモデルによって再現された。[65]」この意味でフィリップスは古典的な大気大循環の成因論に数値実験によって決着をつけた、といわれる。

こうして、数値予報は一九五〇年代に実用化の時代を迎え、一九八〇年代以降は数値予報は天気予報の根幹として揺るがぬ位置を獲得している。ちなみに現在気象庁で運用されている数値予報モデルの一つでは、日本列島を中心とする地域を約二〇キロメートルの解像度で予測し、二日先までの予測計算を約八〇分で完了する。[66]

以上の数値天気予報の例を使って二つのことを論じたいと思う。その一つはモデルの問題であり、もう一つは実験の問題である。そして両者とも法則を用いた予測を行うことが重要である。

さてもちろん、数値解の問題だけならば精度の問題が中心になる。そしてそれは第二節で論じた。シミュレーションで一番の問題は、モデルをどう作るかということである。それも、物理法則によって全てが決まってしまうとはいっても、計算量の問題などもあって、近似をする必要がある。「例えば流体力学方程式を有限の格子で表わすとき、格子以下の現象は直接方程式の差分化で表現し得ないから、その効果を何らかの方法で格子点で与えられる変数を用いて表現せねばならない。格子点での変数をパラメータとして、それより小さい現象、例えば乱流とか積雲対流の効果（運動量・熱・水蒸気の鉛直輸送や凝結による降水と潜熱放出の効果）を数式表現しなければならない。」[67] また別に、「例えば、大規模山岳によって生じる偏西風の蛇行を調べるには、大気力学の方程式は精度よく解かねばいけないが、海水温分布や土壌水分などは考慮しなくてもよい。逆に、CO_2 の増加による気候の変化を実験するには、海水温や土地の乾燥度こそ最も重要な変量であり、大気の運動は全体としての精度以上に詳しく扱ってみても意味がない。」[68] こうして、多様な近似が必要とされるためにモデルは問題により作成者によって異なることになる。ただ、数値実験するためにその分野がある程度成熟する必要があるという点も、リチャードソンの失敗の教訓なのである。

そこで、一九七〇年代の後半あたりから各国の大気循環モデルの性能を比較する試みが行われた。[69] その結果、モデルの定式化は異なっていてもモデルの持つ気候値が相互に共通していたにもかかわらず、自然のもつ気候値とは系統的にずれていることが分かってきた。しかもこの誤差は水平分解能を上げるほどひどくなるというものだった。これは結局は陸面に関する効果だろうとして研究が進められたのであるが、このとき面白いのは数値モデルの組織的研究によって新たな研究課題が生まれたということである。数値モデルは、分かっていることを目に見えるようにしているだけのものではない。

もう一つモデルに関して面白い点は、もとの方程式が解析的方法にもさらにはその当時の数値的方法にも適していなかったのでモデルを単純化したことである。解法の制約にモデルが限定を受けている。流体力学の方程式で一般的

に記述できる現象は音波や重力波などの事象もカバーする。そこで気象学上の現象だけを扱えるように「数学的フィルター」を設けて方程式を単純化した。また以前は図表的方法を用いて視覚的な解法が使われたこともある。解を求め天気を予測するためにはこのような単純化が必要になる。これはモデルを現実と一致するように作っていこうというのとは違った方向である。もちろん実際的な計算の制約があるためだと言うことはできるが、実はこのように単純化しても（つまり理論が現実との同型性から離れても）ある程度の予測ができることは面白い。「モデル化するにあたって計算機の能力の制約から、かなりの条件の単純化、簡略化を行わざるをえない。また、実際には摩擦、衝突、流体の存在、破断など非線形と不確実性が存在する。その解析モデルが適切であり、目的に合ったものになっているかどうかの懸念がつねに存在する」[70]と言われているのだ。モデルに関する以上の論点は、モデルの表現そのものが実在に厳密に対応するとはいえないことを意味している。しかも、シミュレーションしようとするモデルは現実の現象との対応をとるという粗視的な対応が基本となるために、その評価はかなり難しい。

次は実験の問題である。まず、ここで扱おうとする実験のタイプを区別する必要がある。よく哲学で提出されるのは、この現象を説明する法則は何か、それはどのような要因が効いているか、このような点を確認するための実験である。例えば、ガリレオの落体の法則の実験やミルの一致法や差異法はそのような実験を扱っているものと理解できる。このように理想化し純粋化することが目指される実験とは少し違ったタイプの実験が存在する。それは基本的な諸法則が予め分かっているということから出発する実験である。諸法則が微分方程式として予め分かっていても、それらが具体的に相互作用するときに何が起こるかは簡単には分からない。風洞など流体力学に関わる実験はこの典型である。また、設計に典型的なように、要求仕様が決まっていてそれを満たす人工物を構成しようとするときにも実験が必要となる。自動車の衝突実験はその典型である。シミュレーションは風洞や衝突実験に近い。現象を支配する法則についてはある程度の知見が得られているにもかかわらず、様々な実験が必要となるのである。

さて、実験は実物模型で行う場合も、それを小さく扱いやすくした物理模型で行う場合もあり、数値シミュレーションというのはいわば情報モデルを用いた実験だとみなせる。このとき、理論解析と物理実験とシミュレーションとを対比してみると、シミュレーションの特徴が明らかになる。「対象に関する研究が十分に進んでいて、一般的な理論モデルが求まっているときは、対象のふるまいは与えられた環境の下でのその理論式の解を求めることによって得られる。他方、技術開発の初期段階のように、対象に関する知見が非常に乏しいときは、物理モデルによって定性的な傾向を知ることから始めねばならない。」[71] 物理モデルはより進んだ段階でも用いられるが、モデル作成のための時間やコスト面で有利ではなく相似則という壁が存在している。例えば大型の飛行機を対象として実験しようとしても、実物大のモデルは高価であり縮小模型を使うことが多いが、その場合相似則が効いてきて知りたい結果が得られないことにもなる。[72]「計算機によるシミュレーションはこの中間にある。対象のモデルを情報的に表すには、対象の構造に関して、ある程度詳細な知識が必要である。しかし対象の全体としての挙動を表す理論モデルが求まっていないときでも、部分のモデルを相互に関連づけて全体の構造を作り上げ、また部分の挙動の相互の関連から全体としての挙動を求めることができる。これは複雑なシステムの評価には特に適している。それは対象が複雑になるほど、物理モデルの実現も理論モデル化も困難になるからである。」[73]

現在流行している複雑系の科学が対象としているように、非線形の微分方程式で表される現象や多数の要因が相互作用している場合には、結局のところシミュレーション以外にその現象を解明する方法はないかもしれない。第二節でも述べたように、解を求めることには、方程式を書く（つまり現象を記述する微分方程式を作る）こととは違った問題が存在している。現実の天気を予測するためには方程式を解かなければならない。しかも、複雑な熱や流体の相互作用を経た上でどのような現象が起こるかが問題である。[74]

たとえば、素粒子物理学と高エネルギー物理学とはほとんど同一の内容をもつと考えられる。すると、「標準理論」

が確立されると高エネルギー物理学はやることがなくなるかといえばそうではない。標準理論によって高エネルギーの現象が予測できないと、分かったことにならないし、何が予測されるかが明確にならないと、逆に標準理論の改変も提案できなくなる。しかし、きわめて複雑で非線形なシステム（特に量子色力学）を扱わねばならないために、紙と鉛筆では足りずコンピュータでの計算が必要になる。

ここにはモデルの作り方が関わる。しかし、モデルを作る時の問題だけでなく、相互作用の結果を出して、現実の現象と対比する必要がある。このような結果を導くことはつまらない仕事に思えるかもしれない。しかし、計算量の理論を考慮するとそんなに単純な仕事でもないことが理解される。実際、高エネルギー物理学という別の分野が成立しているのである。

また、例えば宇宙物理[75]では、そこで作られた理論モデルの中に関係する重要な物理過程がすべて含まれているかどうか分からないことがしばしばある。このとき、数値シミュレーションを行ってみて、観測が再現できないとしたら差し当たり三つ可能性が残る。つまり、何らかの物理的要因を見逃しているか、数値計算そのものに問題があるか、それとも観測が間違っているかである。このことは、数値シミュレーションによって実験データが得られたことを意味する。

さて、シミュレーションは多くの相互作用を考慮に入れてできるだけ現実と似た状況を構成していこうとする総合をめざす実験を行っている。理論を「演繹的に」理解して、計算して、そこから何が導けるかを考える。それを大規模にやっているだけのようにも思える。ただしこれは人間にとってもコンピュータにとっても自明な演繹的結論を出しているにすぎないとは言えない。ここで重要なのは計算量の問題である。この点を考慮することによって実験による新たな発見の可能性が理解できるようになるのである。

数学においては、一般の決定問題のようにそれを解くアルゴリズムが存在しないことが示されている問題がある。もちろんある種の決定問題は肯定的に解ける。しかし、数学では問題を解くアルゴリズムが存在するかどうか、つま

り「原理的に」解けるかどうかだけが問題であった。「〝解ける〟という確認のみで満足し、実際に解かないならば何の問題もないのだが、実際に解こうとすれば、アルゴリズムの良否、計算の手間が問題になるのは当然のことである。コンピュータが登場するまで、実際に解くことを考えもしなかった問題は数多い。そしてコンピュータの性能が向上し、従来、原理的には解けるが、実際には解こうとしなかった問題を解いてみようと考えたとき、初めて、計算の手間、アルゴリズムの質が問題にされたのである。」[76]つまり、計算量の理論はコンピュータの登場を待って初めて意識されたのである。千倍速いコンピュータができても、例えば巡回セールスマンのような問題（訪問すべき都市と各都市間の距離が与えられた時、各都市をちょうど一度ずつ訪問して戻ってくるのに、どの順に都市を訪問すれば全行程の距離が最小になるかという問題）は実質的に解ける範囲がほんの少ししか広がらない。[77]そのようなタイプの問題が多数発見されているのである。

ここで、シミュレーションのもつ理論的側面と実験的側面を対比しつつ論点をまとめておこう。「計算機シミュレーションは理論とは、現実の物質ではなくモデル系を対象とする点で、また実験とは、対象とする体系の性質を具体的に求めるという点で共通している。そのため、理論に対しては、モデル系の性質を調べる実験的な方法という面を持ち、実験に対しては、完全にモデル化された体系を扱う点で理論的な面を持つことになる。」[78]

実験的な側面は、解析解がなくても、数値を求めることができ、そのものの振舞いを調べられるということに示されている。しかも、シミュレーションの場合は実験条件を自由に制御することができる。従って、高温、高圧、爆発の可能性などの極限状態においても実験を行える。例えばチベット山塊を取り除いた実験をすることによって、それがアジア特有の気候に果たす役割を明らかにできる。[79]このように、実験するときに特定のパラメータを変化させることによって、その影響を調べることができる。しかも、通常の物理実験では測定の可能性に制約されるが、シミュレーションでは数理モデルを使っているために微視的なすべての情報を手に入れることができる。もちろん、モデルを

対象にしているためにそのモデルに対応する現象をうまく表わしているかどうかの判定は重要であるが、実際はなかなか難しい。また、コンピュータの計算能力に制約される。しかも、基本的に数値的な結果しか得られない。数値積分の説明でも述べたように、基本的には特定のある条件の下での特定の数値しか得られない。[80] 従って、一般的な傾向の理解は、理論の解析に比べると弱い、しかし、例えば可視化や動画によって視覚的な仕方でシミュレーションの結果を獲得できる。これは、少し奇妙な仕方ではあるが人間のパターン認識能力を利用した一般性の獲得であり、発見法としてシミュレーションが役立つ点でもある。

シミュレーションによって得た結果を可視化するようなことをしないと、我々にとっては使えるデータにならない。プリンターに打ち出された数値の羅列が与えられても、役に立たないことが多い。[81] これは、広く言えばヒューマン・インターフェイスの問題である。計算で出てきた具体的数値をその後どのように使うかが問題である。可視化といったことをしないと我々がその結果を利用できない。ヒューマン・インターフェイスの面からだけでなく、データの理解という点からも科学的発見の補助としては可視化は重要である。数値シミュレーションを可視化することによって、方程式が含む物理的過程を体験でき、物理的直観を養うためにも役立つ。線形は元来予測しやすいが、非線形であっても人間のパターン認識の能力によってある程度の理解が可能になる。[82] 科学が「左脳」でのみ行われていると見なすことはもちろんできない。

複雑な相互作用はシミュレーションでしか扱いにくいであろう。しかし、その正当化は非常に難しい。全体と全体とで比べるしかない。部分を変化させてある要因がどの程度寄与するかを調べることはできるかもしれない。しかし、その要素の寄与が非線形的な影響を持っていた場合（例えば脳のニューロン）にはそれを予測することは難しく、どういう寄与として理解するかは非常に難しい問題となる。また、一般にシミュレーションできてもどの程度現象をうまく説明しているかは問題である。例えば、人工知能の初期の例としてワイゼンバウムの ELIZA というプログラ

ムは、ある程度うまく人間を模倣していたがもちろん誰もそこに人間と同じ知能が存在するとは考えなかった。それに対して、数値シミュレーションは基本法則が知られた上でそれを組み合わせて何が起こるかを見ようとしている。この意味での構成的方法として数値シミュレーションを使う場合には、（一般のシミュレーションと比べると）それまでの科学の成果を信頼する限り十分信頼のおける方法だと見なせるだろう。もちろんモデルの修正が様々なレベルで何度も行われているとする。

四　計算科学の哲学的問題

シミュレーションは道具としてのコンピュータの特徴、コンピュータによる数値計算に関わる特徴を受け継いだ上で行われている。これらを踏まえた上で、様々な要因が組み合わさったときに何が起こるかを解明しようとしている。つまり、様々な要因を構成することによって、現実の世界に起こることをできるだけそのまま実現させることによって「予測」に寄与しようとしている。ここでまず面白いのは、流体に関する新しい科学法則を見つけようというのではなくて、ある程度確立した法則に基づいて予測をしようとしたことである。法則の確立とは違う予測の試みを行うことも科学的探求の重要な一部である。例えば、理論という普遍的法則と特殊な初期条件から得られる演繹的結論が予測であると言われることがある。[83] しかし、コンピュータによる数値計算の問題やシミュレーションにおけるモデルの簡略化等の条件を考えると、少なくとも「演繹」を厳密な意味で考えるのは間違いだということが分かるだろう。我々が現に予測に用いている知識をあまり単純化して理解すべきではないだろう。法則に初期値を代入すると、予測したいものの位置がすぐに計算できる、ほど単純ではないことを、第二節の数値積分の説明のところで述べておいた。基本法則がどのように正当化されるかという問題よりも、それらの知識を使った人工物の設計がどう行なわれているかの方が、現在の科学技術を理解する枠組みとしては現実に即しているように思える。そのとき、予測に関する理解

もより具体的であるべきだろう。地球温暖化のシミュレーションの評価も、この詳細に依存する可能性がある。

コンピュータを使うことそのことは、数学的に透明な問題ではないことを第一節で論じた。チューリングマシンの具体化だと言う人はいても、そうでない工学的条件が学問性には効いていることを見てきた。いわば幾つかのトレードオフを考慮に入れた上で、実際の計算がうまく行えるように作られたのがコンピュータだといえる。これは、コンピュータサイエンスについて発言するときに論理的観点からのみ哲学的問題設定をするのは少し問題を含むということを意味する。

人工知能の研究から教訓として受け取るべき論点がある。それは、「おもちゃの世界 toy world」と「現実の世界 real world」の区別だ。前者においては首尾一貫して説得的な説明が、後者においても説得力のある説明に移行するとは限らない。これは単に複雑さの量が増すという話ではなく、スケールアップによって本質的に違った対処が必要とされるということである。大きな有限を扱う方法が問題になってきた。これは数学的にまともに扱えない。だからこそ工学的な扱いが必要になる。工学のような予測し設計をする（解を必要とする）実学においては特にこの区別が重要になる。計算量の問題は、具体的に解を必要とする場合には効いてくる。

次には数学の哲学と結びつく問題点である。計算して答えを出す場合においても、数学的に同型であるかの確認だけが応用にとっては問題だとは言えない。同型性なしにでもコンピュータは使える。数値計算と結びついてこの点を概観した。このとき数学の応用可能性が問題である。つまり、1+1=2 という数式とミカンが一個とリンゴが一個を合わせて二個になるということとの関係である。前者は、形式主義によると意味のない形式だ。後者は、ある意味で経験的な命題だ。つまり、両者の同型性が主張できるかということと後者の知覚の記述が曖昧な概念を使用することになっていることが哲学的に問題にされている。[84] しかし、実は応用可能性には、同型性以外に、複雑性の問題が存在する。数学的形式に対して代入するということだけで応用可能性がすべて説明できるとは思えない。ここで問題にし

たいのは、数学の定理があるにして、それを科学技術に応用する際にコンピュータを使うことが多い。コンピュータは有限の機械であり、モデルの簡略化も行われている。にもかかわらず、応用が可能であると考えられている。このことは数学の応用可能性の根拠について同型性以外の論点も重要だということを示しているように思える。初等数学の法則を満たさない。このようなものを使いながらもコンピュータは役に立っている。有限性という問題を含んだ機械でありながら、コンピュータは役に立っている。この点が、面白い点である。

また、数学の知識は、証明との関わりで問題にされてきた。厳密な学問としての数学である。しかし、もともと数学は、計算をするという意味で昔から使われてきた。その面を問題にしてきた。コンピュータを使う計算をすることは、その有限性という制約が表面に現れるために、可視化という方法を使って人間のパターン認識の能力に頼ったり、プログラムの正当化に工学的な安全性の基準を使ったりしている。シミュレーションの発見法としての役割について取り上げたように、人間的な知的能力の重要性も数学の応用と結びついて問題にされているのだ。

以上幾つか示唆してきた点について、少し一般的な言い方で述べれば、コンピュータを使って世界を予測しコントロールするときには、世界の機械化や幾何学化が進んだということの指摘ではすまない問題があるということを見てきたのであった。世界をコントロールしようという「意図」を強調したい人があっても、意図とその実現との間に大きなギャップがあるとすれば、その関連をまず解明することが必要だろう。

註

(1) Computational Science（計算科学）という分野は新しい。計算科学は、一九八六年に Kenneth G. Wilson（ノーベル物理学賞受賞）によって、米国の大規模科学技術計算研究の不可欠なツールとして提唱された（「計算科学への期待」田子精男、『機械の研究』第四九巻第一号（一九九七）七八頁を参照）。日本でも同時期「計算理学」という学際的な学問分野を立ち

上げようという試みがあった（「序文」上村洸、『別冊数理科学　シミュレーション』サイエンス社（一九九三））。なお、一九九八年四月の時点で YAHOO! USA を検索すると、Computer Science という分類の下位に Computational Science という分類が存在し、その中に一八の関係項目が含まれている。それに対して「コンピュータサイエンス」という分類をしている YAHOO! JAPAN では対応する下位の分類は存在せず、「計算科学」で検索すると一件だけ関係するページが見つかった。ちなみに、より特殊な分野である計算物理学の専門雑誌 Journal of Computational Physics は一九六六年に刊行されていた。

（2）　コンピュータは科学計算以外にも様々な側面から取り上げられ分析されている。第一に、ディジタル処理をしているとか、記号処理ができるとかいう側面がある。これに関しては、記号や言語を処理する機械という側面が取り上げられている。第二に、顧客名簿の流出などの通信に結びつく情報処理やその社会的問題にも関わる。これは第一の問題とも結びついて、インターネットに関する法律的倫理的問題をも提起している。そして第三にコンピュータがどのようにして発明され発展したかという起源や特許に関わる問題も取り上げられることがある。そして第四に、現代では洗濯機や自動車をはじめ様々なところにマイコンが使われている。このような制御機械としてのコンピュータの社会的問題も大きい。その他様々な側面からコンピュータについての哲学的考察が行われている。

（3）「計算物理学と CP-PACS 計画」岩崎洋一、宇川彰、梅村雅之、『情報処理』Vol. 37 No. 1,（一九九六）一一頁。なお、この段落の論点は、この論文による。

（4）　車社会に移行したときのインパクトやグーテンベルクの印刷術による影響と類比的な影響力をインターネットは我々の文化に与える可能性はあるだろう。つまり、スピードや量の飛躍的増大によって社会的に大きな影響を及ぼすことがある。そして、その結果として生じた車を含む社会システムや印刷文化が注目されるようになった。しかし、コンピュータと車や印刷術のテクノロジーとの本質的な違いは、前者が汎用的な機械であって、後者のように一つの（もしくは少数の）目的にのみ使われるのではないことにある。だから、車の構造を知ることは特に必要だとは思えないにしても、コンピュータは汎用の機械であるために機械そのものを更に問題にする必要が生じる。この論点とは別に、インターネットが物理学研究者に重要な役割を果たすという主張については「計算革命と物理学研究」アルフレッド・ブレナー、『パリティ』Vol. 12, No. 07（一九九七）、

"The Computing Revolution and the Physics Community" Alfred E. Brenner, *Physics Today* (1996-10) を参照。

(5) カリキュラムを手がかりとしたコンピュータ・サイエンスの学問性については、以前論じたことがある。拙論「コンピューティングの工学化」、『大阪体育大学紀要』第二六巻（一九九五）二一一―二三〇頁。

(6) 実はアナログではなくてディジタルな機械であるということによって、普通の機械に見られる精度の限界や速度の限界のような現実的な技術上の問題は無視できることになった。つまり、普通の機械は、表面の仕上げや歯車の切削といった工作の精度には実際上の限界があるばかりでなく年月が経つと部品の精度が落ちて劣化する。また機械の運転速度が速くなると精度が落ちることが多い。しかしこの意味の精度からは独立にコンピュータの精度を決定できる。（この点の指摘は『計算機の歴史』ハーマン・H・ゴールドスタイン　末包良太・米口肇・犬伏茂之訳　共立出版株式会社　一五七頁以下による。）もちろんこれだけのことなら算盤と同じようなものである。この点も含めてコンピュータは機械や道具としては面白い性質をもっている。

(7) もちろん、コンピュータは機械なので最終的には物理的影響を無視できない。その点については例えば、『岩波講座情報科学1　情報科学の歩み』高橋秀俊（一九八三）の第六章、また「情報科学への招待」（特に第二章）山田真市、『コンピュータから生まれた新しい数学』野崎昭弘・廣瀬健編　別冊数学セミナー　日本評論社（一九八六）などを参照。

(8) 「プロローグ」國井利泰、『bit 別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版　一九九三年一月 iv 頁。

(9) 「カリキュラム68」、『bit 別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版　一七頁、"Curriculum 68" *Communications of the ACM*, Vol. 11, No. 3 (1968) p. 161（以下、邦訳を提示した場合は、特に断らない限りその訳文を使用した。）

(10) 『大学等における情報処理教育のための調査研究報告書』社団法人情報処理学会　大学等における情報処理教育検討委員会　三一頁、平成三年三月。

(11) 「カリキュラム91」、『bit 別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版　一七二頁。

(12) 『岩波情報科学辞典』（長尾真等編）によると、情報科学 computer and information science とは、「情報の生成、伝達、

変換、認識、利用などの観点からその性質、構造、論理を探究する学問、およびその具体化を行う計算機を中心とする情報機械のハードウェア、ソフトウェアの理論と実際に関する学問の分野をいう」と規定されている。また、情報工学 information engineering は、日本だけで用いられる語であって、「情報工学と情報科学との間にあまり差がなく、ほとんど同義的に使われる」と述べられている。また、計算機科学 computer science は、「情報科学の一分野。計算機に関する理論、ハードウェアおよびソフトウェアに重点をおいた分野をいう。情報そのものではなく、情報を収集し、伝達、蓄積、加工する枠組みとしての情報機械を研究対象の中心とする。」ただし、計算機科学と情報科学との差は縮まりつつある。また、コンピュータサイエンス、コンピュータエンジニアリング、情報学その他の類似する学問をまとめて、コンピューティング（計算機分野と訳されることもある）と呼ぶこともある。

(13)「情報処理教育における実験・演習」都倉信樹、『情報処理』Vol. 32, No. 10, 一九九一年一〇月参照。

(14)「情報処理教育における実験・演習」都倉信樹　一一〇二頁。

(15)「同論文」都倉信樹　一一〇三頁。

(16)「同論文」都倉信樹　一一〇三頁、また、コンピューティングにおいては analysis よりも synthesis が中心だという論点については、「情報処理専門教育の一実例」小谷善行、高橋延匡、『情報処理』Vol. 33 No 2,（一九九二）も参照。

(17)「数学や物理学のコースでは、典型的な一つか二つのやり方しか、問題解決のし方がありません。そして問題解決のし方は文字どおり、以前の何百万人の学生と同じかたちで解決していくわけです。しかしコンピュータ・サイエンスにおいては、二人の人が同じ問題を、同じやり方で解決するということはありません。自分のアイディアを使ってどのように問題を構造化し、解決していくかを決めていくわけです。問題は大ざっぱに記されているだけです。細部には言及されません。ですから、学生たちは問題解決の前に、問題を具体的なものにしていかなければなりません。それがコンピュータ産業の特徴なわけです」（四八二頁）「創造科学としてのコンピューティング――障壁を取り除く」アンドリース・ヴァンダム、『bit』Vol. 22, No. 5 一九九〇年五月。なお、bit のこの巻では「コンピュータサイエンスにおける創造性」という特集を組んで、ACM カリキュラム88をめぐる議論を展開している。

(18) "Computing as a Discipline" Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young, *Communications of the ACM* Vol. 32, No. 1 (1989) p. 10. これは拙訳である。ただし、「学問としての計算機分野」ピーター・デニング等木村泉訳『情報処理』Vol. 31, No. 10 (Oct. 1990) と『コンピュータサイエンスのカリキュラム』國井利泰編　共立出版を参照した。

(19) "Computing as a Discipline" Peter J. Denning et. al. p. 10 拙訳。

(20) "Computing as a Discipline" Peter J. Denning et. al. p. 10f.

(21) 『大学等における情報処理教育のための調査研究報告書』社団法人情報処理学会　大学等における情報処理教育検討委員会　平成三年三月　一〇五頁以下。

(22) 『同上書』一〇五頁。

(23) 『同上書』三四頁以下。

(24) 『大学等における情報システム学の教育の実態に関する調査研究』社団法人情報処理学会　大学等における情報システム学の教育の実態に関する調査研究委員会　平成四年三月　二六頁以下。

(25) モデルは対象との一致だけが求められているのではない。人工知能の研究とも結びついて、問題解決において問題の表現をどう決めるかによって解決に要する時間などが変わってくることが指摘されている。

(26) 例えば、木村英紀は「制御とモデル」、『計測と制御』第三七巻第四号（一九九八）において、モデリングの重要性と難しさを述べ、「モデル」学の確立を提唱している。

(27) 『学問としての計算機分野』ピーター・デニング等　一三六〇頁、"Computing as a Discipline" Peter J. Denning et. al. p. 16.

(28) 根本的な問題はプログラムとは何か、という問題だ。コンピュータでプログラムを走らせるということは、対象世界を計算機世界に置き換えた上での操作になっている。まず実世界を機械にマッピングすることが必要である。いわば、実世界から人工界に写像をして、人工物の世界でどのような振舞いをしているかを見ようとするのがコンピュータによる解析やシミュレ

ーションということになる。その上で、うまく計算を行う。ダイナミックな論理であることも問題を複雑にしている。プログラムは何を表現しているのか。実在を表現するとはどういうことか。こういう問題が基礎にある。しかし、これは数学的表現を含めたすべての表現に関わる問題である。"The foundations of artificial intelligence" edited by D. Partridge and Y. Wilks, Cambridge U. P. (1990) においては、人工知能の研究を一種のシミュレーションとみなすことによって、プログラムやその正しさの問題について様々の側面から論じられている。しかし残念ながらここでは論じる余裕がない。

(29) Edsger W. Dijkstra, David L. Parnas, William L. Scherlis, M. H. van Emden, Jacques Cohen, Richard W. Hamming, Richard M. Karp, Terry Winograd: "A DEBATE on teaching Computing Science", *Communications of the ACM*, Vol. 32, No. 12, 1989, pp. 1397-1414,「計算科学教育に関する討論」(前編) エズガー・ダイクストラの問題提起、『bit』1991/4 七七二—七八三頁、「計算科学教育に関する討論」(後編) ダイクストラの主張への同僚の応答『bit』1991/5 九〇三—九一六頁。

(30) "A DEBATE" Dijkstra, p. 1400.『bit』1991/4. 八頁。

(31) "A DEBATE" Dijkstra, p. 1402.『bit』1991/4. 一〇頁。

(32) "A DEBATE" Dijkstra, p. 1403.『bit』1991/4. 一四頁。

(33) "A DEBATE" Emden, p. 1408.『bit』1991/5. 一一六頁。

(34) ibid.

(35) "A DEBATE" Hamming, p. 1409.『bit』1991/5. 一一八頁。

(36) "A DEBATE" Hamming, p. 1410.『bit』1991/5. 一一九頁。

(37) ibid.

(38)「抽象数学に成果をあげている大数学者が工学における難問に解答を与えてくれるには、数学以前の工学的問題を理解してかからなければならないが、これは大変な努力と時間を必要としよう。逆に工学の難問を解こうとしている人々が近代数学を理解して何かの手がかりにしようとするのも望み薄であろう。それは工学上の難問は数学的には汚い問題で、数学も線形の

問題なら素晴らしい数学的成果があるが、非線形の問題はたいていどうにもならないのに似ている。」七七頁「情報工学教育のゆくえ」室賀三郎、『bit 臨時増刊　情報工学の教育・研究』一九八〇年一二月。

(39) "An Axiomatic Basis for Computer Programming" C. A. R. Hoare in *CACM* Vol. 12, No. 10 (1969) p. 576.

(40) この言葉は例えば、"Program Verification, Defeasible Reasoning, and **Two Views of** Computer Science" Timothy R. Colburn, *Minds and Machines* 1 (1991) p. 97 を参照。

(41) "Program Verification: The Very Idea" *Communications of ACM* Vol. 31, No. 9, (1988) pp. 1048-1063. なお、彼自身は論争の最終のまとめを次の論文で行っている。"Philosophy and Computer Science: Reflections on the Program Verification Debate" in "The Digital Phoenix: How Computers are Changing Philosophy" edited by Terrell Ward Bynum and James H. Moor, BLACKWELL (1998).

(42) この論点は、〔Technical Correspondence〕W. R. Bevier, M. K. Smith, and W. D. Young p. 375 *CACM* Vol. 32, No. 3 による。

(43) 「情報科学への招待」山田真市、『コンピュータから生まれた新しい数学』野崎昭弘・廣瀬健編　別冊数学セミナー　八頁、日本評論社（一九八六）ここでは計算量の理論の成果として、証明を機械で行う場合の障壁が示されている。

(44) 『同上書』一〇八頁。

(45) 『ノイマンとコンピュータの起源』ウィリアム・アスプレイ　杉山滋郎・吉田晴代訳　一一三頁、産業図書。"John von Neumann and the Origins of Modern Computing" William Aspray, MIT Press (1990) pp. 108f.

(46) この段落の論点に関しては、「情報処理基礎理論」福村晃夫・本多波雄、『bit 臨時増刊　情報工学の教育・研究』坂井利之他編　一九八〇年一二月　共立出版　三一頁を参照。

(47) ここで説明しているのは非常に単純な力学系である dx/dt＝f(t) という微分方程式を念頭において説明している。もちろん、f(t) が連続であるとかその他数値計算上の種々の制約については無視して、基本的なイメージだけを提示しようとしている。

(48) 『岩波講座　応用数学　微分方程式の数値解法 I』三井斌友　五頁。

(49) 『計算機の歴史』ハーマン・H・ゴールドスタイン　末包良太・米口肇・犬伏茂之訳　共立出版株式会社　三三四頁の訳者の注を参照。

(50) その他、実数としての性質や自然数に関するペアノの公理など基本的な数学的性質に関して、数学とコンピュータでは異なっているということが次の論文でうまくまとめて書いてある。「計算機とその数学的理論について」島内剛一、『コンピュータから生まれた新しい数学』野崎昭弘・廣瀬健編　別冊数学セミナー　日本評論社（一九八六）一六一―一六四頁。

(51) 『前掲書』一一二頁。ウィリアム・アスプレイ。Aspray, pp. 98f.

(52) 『大学等における情報システム学の教育の実態に関する調査研究』社団法人情報処理学会　大学等における情報システム学の教育の実態に関する調査研究委員会　平成四年三月　一九頁。

(53) 『前掲書』、ゴールドスタイン　一五九頁。"The Computer from Pascal to von Neumann" Herman H. Goldstine, Princeton U. P. (1972) p. 142.

(54) 『前掲書』三井斌友　八頁以下。

(55) 「算法的思考と数学的思考」(『クヌース先生のプログラム論』有澤誠編　共立出版）という論文で、クヌースは両者の相違を「操作の節約」という意味での関心が数学にないこと、および「過程の状態に関する動的な概念」である「指定演算 assignment」が数学にないことだと述べている。数値計算においては計算結果に特に注目しているために、後者の論点はこの小論では特に取り扱っていない。

(56) 本来なら独立であるはずの要素が、たまたま特定の値をとったために互いに従属になってしまう退化（縮退）も、アルゴリズムにおいて対策のほどこしにくい現象である。

(57) 「高エネルギー物理学におけるスーパコンピューティング」小柳義夫、『情報処理』Vol. 36 No. 2,（一九九五）一三五頁。

(58) この段落の情報については、「温帯低気圧モデルの歴史的発展」岸保勘三郎、『天気』Vol. 29 No. 4（一九八二）による。

(59) 以下のリチャードソンの数値予報に関する情報については、「温帯低気圧モデルの歴史的発展」岸保勘三郎、『岩波講座　応

用数学　計算理学の方法』能勢修一、寺倉清之、松野太郎、佐藤信夫　七七頁以下、および「天気予測――その発展のあしどり――」永沢義嗣、『日本機械学会誌』1997.1 Vol. 100 No. 938, 七九―八一頁による。

(60)　時間積分する場合に、その積分時間間隔が、格子間隔を最大の影響伝達速度で割った商よりも小さくなければならないといけないという条件。格子間隔を短くすると、より詳細な様子が分かる。しかし、格子間隔を短くすると、時間間隔も短くしなければならない。一般に、格子間隔を二倍にすると計算時間は八倍かかる。リチャードソンはこの線形不安定に関する問題の存在に気づいていなかった。CFL 条件（時にはクーラント条件と言われることもある）については例えば、『前掲書』能勢修一他　一〇〇頁を参照。その条件が示された原論文（CFL は以下三人の頭文字をとっている）は R. Courant, K. Friedrichs, and H. Lewy, „Über die partiellen Differenzengleichungen der Mathematischen Physik", *Math. Ann.* Vol. 100 (1928-29), pp. 32-74 である。また、時間間隔を短くしても発生を抑えられない非線形不安定と呼ばれる現象も存在する（「気象・環境科学におけるスーパコンピューティング」住明正、『情報処理』Vol. 36 No. 2, (一九九五)　一六〇頁以下）。

(61)　『前掲書』能勢修一等　七八頁。

(62)　「数値的アプローチが実用化されるに先立ち、四つの領域で改善が必要だった。方程式を実際に解ける程度まで単純化するために、理論家たちはそこに含まれる多数の流体力学および熱力学の因子のうちから、天候に最も大きなインパクトを与える因子を選び出す必要があった。また、最も単純化された方程式系でさえ、解析的方法ではとても歯が立たなかったので、数値的な手法やグラフを利用する手法が近似解を得るために導入された。方程式に初期条件と境界条件を与えるために、膨大な観測データを体系的に集め、共通の単位に換算する必要があった。最後に、そしてこのうえもなく決定的なことであるが、毎日なされるたった一回の天気予報に必要な数百万回の算術計算を行うために高速の自動計算機が必要であった。」（『前掲書』アスプレイ　一三六頁以下 Aspray, p. 123.）もちろん、気象学の確立のためには、計算時間は無関係だとして「原理的」な問題だけを考えればいいとされるかもしれないが、実際の天気予報を行うときにはそうではない。そのため、解析的方法がない場合には図表的方法を用いたりして解を求めようとしてきた（『同上書』アスプレイ　一三九頁以下参照 cf. Aspray, p. 125f.）。

(63) その後、バランスモデル、プリミティブモデル、全球スペクトルモデルというように、モデルが複雑になり精度も向上してきた。「数値地球流体力学序論」住明正、『別冊数理科学　シミュレーション』サイエンス社（一九九三）八一頁参照。

(64) 「コンピュータ・シミュレーションと現代気象学」松野太郎、『別冊数理科学　シミュレーション』サイエンス社（一九九三）七三頁。

(65) 「同上論文」松野太郎　七三頁。

(66) 計算機による二四時間予報に要する時間（三〇〇キロメートル格子ではあるが）は、ENIAC の場合は二四時間だった。そして一九五三年六月三〇日までに六分間に短縮された。（『前掲書』ゴールドスタイン　三五〇頁を参照 cf. Goldstine, p. 304.）。

(67) 『前掲書』能勢修一等　八六頁。

(68) 「前掲論文」松野太郎　七四頁。

(69) この段落の情報については、「前掲論文」住明正　八二頁以下を参照。

(70) 『設計という名の問題解決』黒川篤　オーム社（一九九七）八八頁以下。

(71) 「知識処理による問題解決の高度化へ」大須賀節雄、『日本の科学と技術九―一〇　シミュレーション』Vol. 27, No. 241,（一九八六）一〇〇頁以下。

(72) 「前掲論文」松野太郎　七八頁。

(73) 「前掲論文」大須賀節雄　一〇一頁。

(74) この段落の論点は、「高エネルギー物理学におけるスーパコンピューティング」小柳義夫、『情報処理』Vol. 36. No 2,（一九九五）一三二―一三六頁を参照した。

(75) この段落の論点は、「コンピュータで宇宙の神秘を探る」マイケル・ノーマン、『パリティ』Vol. 12 No. 8,（一九九七）による。

(76) 「はじめに」廣瀬健、『コンピュータから生まれた新しい数学』野崎昭弘・廣瀬健編　別冊数学セミナー　二頁、日本評論

社（一九八六）。

(77) 例えば、「問題の難しさの計量」谷口健一、『計算の効率化とその限界』伊理正夫・野崎昭弘・野下浩平編著　数学セミナー増刊、日本評論社　九九頁を参照。

(78) 『前掲書』能勢修一等　三頁以下。

(79) これらの例に関しては、『同上書』能勢修一等　八〇頁以下を参照。

(80) 以上のような論点に関しては、『同上書』能勢修一等　四頁を参照。

(81) この論点は多くの著作で指摘されていることであるが、例えば「材料科学研究におけるスーパコンピューティング」川添良幸、『情報処理』Vol. 36. No. 2.（一九九五）一五二頁以下を参照。なお、可視化を扱う「可視化情報学会」の学会誌は、一九九八年四月号「小特集　模型表面流れを可視化で読む」が既に Vol. 18, No. 69 に達している。

(82) 「第三の研究法――数値実験」カールハインツ・ウィンクラー、ジェイ・チャルマーズ、ステファン・ホッドソン、ボール・ウッドワード、ノーマン・ザブスキー、『パリティ』Vol. 03, No. 07,（一九八八）三頁参照。

(83) 例えば、『客観的知識』カール・ポパー　三九二頁以下、木鐸社。"Objective Knowledge" Karl Popper p. 352f. Oxford U. P.

(84) 例えば、『数学の哲学』ケルナー　一五四頁以下、公論社。"The Philosophy of Mathematics" Stephan Körner, pp. 106f. Dover を参照。

（筆者　さいとう・のりふみ　大阪体育大学体育学部〔哲学〕助教授）