# 数　値　実　験
## ～学問的意義と実験的側面～

春海佳三郎*

### 1．はじめに

数値実験またはコンピュータシミュレーションという言葉はスーパーコンピュータを用いた大型数値計算という意味に用いられることが多い。私の専門の弾性波のシミュレーションはまさにそうした大型計算に属している。その結果は予想外に様々な実験事実の解明に役立つことがわかったが、余りにも専門的で一般の方には馴染まないと考えた。そこで数値実験という言葉を広義に解釈して、数値計算の実験的側面も含めた形で説明することにした。最後に、情報教育に対する筆者の意見を述べさせてもらった。

### 2．理論と実験

他の学問分野におけると同じように理工学の分野でも、理論と実験の２つの分野があって、互いに助け合って各分野の理解と進歩がなされていると考えられている。

実際には、学問分野にもよるが、理論的研究者と実験的研究者とは、相互に批判しあいながら仕事をしていることが多いのではなかろうか。

もちろんこれは平均的な、もしくはそれ以下のレベルの話かもしれない。理論的研究者にしても実験的研究者にしてもノーベル賞をもらうような人々は賞賛され、評価されているに違いない。しかしこのような人は何十万人に一人、特にわが国ではその比はさらに低くく、平均的な人が沢山いるというのが実状だろう。

アインシュタインですら、ある部門では現在になってみると間違っていたことはよく知られた事実である。それ故、他人の仕事を批判するだけのことならば赤子の手をひねるよりも容易であり、理論的研究者、実験的研究者ともに相手を批判するなどということは実は誰でもできることなのである。

理論的研究者のもっとも陥り易い欠点は、「自分の仕事以外は皆つまらなく、自分はそんなつまらないことには興味がないのだ」という態度または言動をとることである。一方実験的研究者は「理論屋は何もわかっちゃいない。その証拠に、何一つ自分の疑問に答えてくれない」といって嘆いている。このように、理論的研究者が「難しいことは価値があり分かりやすいことはつまらない」というのは、誤りである。私の経験ではむずかしいことのみが価値があるという考え方は、学生時代に陥り易い誤りである。自分以外の人に理解できない理論は、結局誰も相手にしてくれないことになる。

一方、実験的研究者には理論ですでに得られていることから自分の欲している結論をある程度は自ら導き出すぐらいのことはして欲しい。自分で努力をしないで、「棚ボタ」式に都合のよいことのみを希望するのは困りものである。

しかし現実には、マクロ的には理論も実験も着々と進歩しているように見える。特にエレクトロニクスの進歩によって、実験の分野の進歩に従来の理論が追いついて行かなくなっていることも事実である。

近年の様々な学問分野での発達、発見にはコンピュータが大きく関わっている。宇宙物理・流体力学・材料科学・高分子化学等の、純粋科学から工学までの広い学問分野でその多くの事例が見いだせる。人文系の研究においてさえ例外ではないようである。

* 東京情報大学教授

私の関係している超音波の分野でも、コンピュータを利用して現象に対する理解を深めたり新たな知見を得るようになってきている。これはコンピュータあるいはマイクロプロセッサーの性能の向上が著しい（価格は変わらないで）ため、計測・データ処理等の場面においてハードウェアーを従来以上に手軽に利用することができるようになった（この事情は工学の各分野でも同様で我々の身近な機器を見ても容易に想像がつく）ことが重要な要因である。

さらに通常の時間・空間スケールでは我々人間の知覚では把握できないような現象を、コンピュータによって数値的に現象を再現する（即ちシミュレート)ことができるようになった(例えば原子炉の圧力容器に物体が衝突したときの破壊の様子、太陽内の電磁流体現象の解明等)。このことにより、人間にとって理解し易い形で現象を表すことが可能になったのもハードの容易な利用といったこと以上に重要である。

すなわち、人間が自然現象を理解するために従来からあった理論と実験の２つのツールの他に、新たに数値実験というもう一つのツールを人間は手にいれたと言えるからである。コンピュータの影響は従来の理論と実験という分類を変えさせるに到ったのである。こうしてみると、この数年のコンピュータの進歩は学問的な進歩に大きく貢献してきたといえる。

数値実験（Numerical Experiment）なる言葉は20年ほど前には殆どないがごときものであった。数値計算と言われ、理論の結果つまり解析的に得られた解を数値化するためにあるような、ごく小さな極めて受動的な分野として存在するのがせいぜいであった。それがコンピュータの劇的な進歩のおかげで、アクティブな分野へと変貌を遂げたのである。

つまり従来のやり方（解析的に解を求めそれに数値を代入して現実の問題に対する解を得る）では、グラフ一本書くのになん年も、あるいはそれ以上にかかっていたが、今日ではるかに短い時間で、はるかに現実の場に即したデータを出すことが可能になったのである。

## ２．１　理論の役割

理論の分野では２つの主要な役割がある。それは、１）定式化或いはモデリング、２）解の導出、である。

1)の「定式化」には様々なレベルがあるが、要するに現象を記述するできるだけ汎用性のある、つまり基礎的な方程式を見つけることである。古典力学のニュートンの方程式・量子力学のシュレディンガー方程式・流体力学のナヴィエストークス方程式等の導出がこれに当たる。現状では、マクロな現象はすでに導出されている古典物理の方程式で記述でき、分子、原子のレベルの現象はシュレディンガー方程式で記述でき、聞くところによると原子核内部レベルのミクロな現象も、ある式でだいたい記述できるそうである。従って身の回りのほとんどの現象について、定式化は終わっていると見なせるだろう。

一方「モデリング」は、すべてを考慮していてはとても定式化できないような複雑な現象、或はオリジナルの定式化がいくら正しくても解が求められそうにないような場合に、適当な仮定をおくことでうまくつじつまを合わせることである（「うまく」つじつまを合わせないと単なる恣意的なものになってしまう)。定式化の際の厳密性をある程度犠牲にする作業ともいえる。厳密性を犠牲にした以上おのずとその適用限界が生ずるわけでそれを知ることが重要なこととなる。

もっとも定式化も一種のモデリングと言えなくもない。連続体力学の基礎方程式なども分子原子レベルの話をしている人にとってはとんでもない大ざっぱな式に映るかも知れない。2)の「解の導出」は、上述のプロセスに依って得られた式を、現象に対応する条件の下で解くことである。定式化が正しいかどうかは、実はこの解を求めると言うプロセスによってのみ確認可能なのである。

従来、これは難しい式をひねくりまわすテク

ニックを駆使して、紙と鉛筆と忍耐と頭脳だけで行なっていた作業であり、労が多い割には限られた系についての解しか得られなかった。今日でもこのような作業の意義は十分あるのだが、凡人がおいそれと手を出せるようなことでもなく出したところで、よほどの運がないかぎり役立つ結果はでてこないことが多い。

今日では、解析的な解（厳密解ともいわれる）は数値解の検証に用いられることが多いのが現状である。そのため今日では前述したようなコンピュータを用いて数値的に解を求める作業が流行となってきている。

### 2．2　実験の役割

従来実験は理論によって予測された現象の確認、あるいは新たな現象の発見とそのメカニズムの解明と言った目的を持って行なわれてきたように思う。現在では、数値計算の大きな進歩によって実験のあり方も変わってきている。

数値計算におけるモデル化に必要なパラメータの決定を行なうためのデータの収集、数値計算の結果の確認といったことが実験の目的として加わってきた。従来からの理論と実験との関係においても、実験はこのような役割を果していた。

しかし数値計算の進歩によってより広範囲の問題が解かれるようになった現在、それにともなって測定すべき対象、測定すべき量、測定すべき範囲等が大きく広がり、実験がなすべき仕事が著しく拡大したと言えよう。こう言った実験に対する要求の拡大に応じて、エレクトロニクス技術の進歩にともなう測定方法の大幅な改良、あるいは新たな測定原理の導入といったことが盛んに行なわれてきている。と同時に、このような実験上の進歩のスピードに従来の解析的方法では応じきれず、現象の解明に数値計算を用いざるを得なくなるといった相互作用的な面もある。

## 3．数値実験について

実験がエレクトロニクスの進歩によって画期的な進歩を遂げたのにひきかえ、理論が解析的な方法に固執していると、実験の要望に答えられないようになってきた。これは実験はマスプロダクション可能なのに対し、理論はハンドメイドでやっているからである。そのギャップを埋めるものとして数値実験がコンピュータを利用して登場していきた。

数値計算は、現在では理論によって与えられた式から解を導き出す最もポピュラーなツールになり、学問的な一分野を構成するようになった。計算物理学・数値気象学・計算力学・計算化学・数値流体力学といった分野が急速に発展してきた（十把ひとからげに計算理学などという人もいるようだ）[文献　1－4]。

これは何度も述べたように、コンピュータの大幅な進歩に伴って解を求め得る範囲と量が従来の解析的な手法とは比べものにならないくらい拡大し、成果をあげてきたことが大きな要因である（事実この種の国際学会が毎年数多く開催されており、又普通の学会でも数値実験の報告が急速に増大している）。と同時に数値実験には理論とも実験ともつかないところがあって半ばハイブリッドな感覚が必要であることも理論、実験と並ぶ分野になった原因であろう。

数値計算によって解を得る作業は理論的なものに映るかもしれないが、数値実験とも言われるように実験的な色彩もかなり強く、一見大したことが無いような所に気をつける必要が往々にしてある。特に大規模な計算ほどそうである。即ち、既存のプログラムに数値を入れて計算しただけでは時としてとんでもない結果になってしまう。数値計算の落し穴といった例をいくつか後に示し注意を喚起し、実験的側面の例を示そう。

### 3．1　理論的な側面

数値計算上の理論とは計算安定性に関するも

の、誤差論等の数学に関連したものから、解こうとしている物理方程式の性質に関連したことまで、多岐にわたる。特に前述したような理論的研究者の仕事であるモデリングは、解を得ようとしている現象が複雑であるほど重要な問題である（本来理論屋さんがやってくれれば有難いことなのだが興味を示してくれるとは限らないので、勢い計算屋が手を出すということになる。もっとも、理論屋さんがやってくれたとしてもその適用範囲等について十分理解しておかないと、とんでもないことになる）。

即ち現象が複雑である場合、変数が多くなったり、時間空間的なスケールが計算対象としている領域に比して小さかったりするために、計算時間、計算に必要なメモリー等の制約から、従来の定式化では計算できないという事態に陥ってしまうのである。例えば乱流計算ではナヴィエストークス方程式をそのままの形で解くことは不可能であり、なんらかのモデルを導入することにより、大きなスケールの量のみを計算する必要があるということである。

特に工学的な分野では、計算時間、メモリーの制限は切実であり、必要なデータは限られたものであることから、これらのボトルネックをクリアーするためのモデリングはますます重要である。このようなモデルの代表としては材料力学の単純梁理論、あるいは回路学の線型回路理論、流体力学の乱流粘性の導入等があげられよう。現在も、複雑なモデルの陰でこれらの単純なモデルが、現象に依っては十分有効であるため、広く用いられている。しかしすでに述べたように、この際にモデルの適用限界を十分理解して利用することが大切なこととなる。と同時に、モデル化抜きの方程式を直接数値的に解くことでモデルの限界を検証するといったこともなされており、理論へのフィードバックといえる。

計算機の進歩と共に、より複雑なモデルによる計算も可能になってきている。例えば、航空機等の設計があげられよう。従来、航空機あるいは大型構造物の設計は簡単な単純梁に近いモデルで計算されていたが、計算機の進歩とともに次第に連続体に近い有限要素モデル（Finite Element Model）を用いて計算できるようになってきた。これに依って強度試験を簡略化でき、さらに限界設計（安全係数が1に近い値）に近い設計が可能になった。

その身近な例としてジャンボ機がある。ジャンボ機の翼が大きく振動しながら飛んでいるのをご覧になった方も多いと思うが、これなどは大型計算による設計の大きな成果と言えよう。ソ連がこのような経済性が重視される商用航空機の分野で、ジャンボ機に比肩し得る機体を長いこと製造できなかったのは、計算機関連技術の大幅な遅れのためといえるだろう。

計算機の発展とともに複雑なモデルを用いた計算が可能になるということは、従来の単純なモデルにつきまとう限界を考える必要が少なくなったことを意味する。同時に、そういった計算が次々と現われてきたということは、現象に対する本質的な支配方程式を仮定抜きで数値的に取り扱う必要が、工学的な分野においても増してきたことを意味する。

即ち、今日のように高度技術の開発が望まれる状況下でこそ、扱っている現象に対する深い理解が必要だということである。技術の猿真似をしたく無ければ、又させてくれない状況になってきた以上、一層基礎的な現象に対する理解が必要とされている。高度技術の集大成である計算機は、それ自体が高度技術を生むための最も強力なツールと考えられる。日米摩擦、あるいはココム等にみられるハイテク分野における様々な規制は上述したことが社会的、政治的レベルにおいても共通の認識となっていることの表れであろう。

### 3．2　実験的な側面

上記のように、数値的に問題を解く際には改めて方程式を導出する必要がある場合もあるが、ほとんどの場合は解くべき方程式は明確になっていて、それを色々な数値的な方法で解き、理

論による値、実際によって得られたデータ、あるいは他の計算結果と比較をすることで信頼性のある解を得ること（つまりは信頼性のあるソフトを書くこと）が必要となる。

このような計算は通常、大型計算機（メインフレーム）或はスーパーコンピュータによって行なわれるものと考えられてきたが、最近では価格の割に高性能のワークステーション（Work Station）によって可能になってきた。このため数値計算は一層身近なもの、誰でもやろうと思えばできるものになってきている。とは言えそう安々とできるわけでもない。特にパッケイジソフトを使わずに自分でプログラムを書くときはそうである。

数値計算を行なう上で入門者が馴染めない一つの要因として、常に有限桁で計算が行なわれる、ということが挙げられるのではないだろうか。学校で教わる数学と違って、数値計算では桁数が有限なため、計算の度に「丸め誤差」が入ってくる。現在ではパソコンでも浮動小数点計算を数秒間に $10^6$ 回行なうことが可能であるから、数値解析の本によく書いてあるように、a＋bの誤差｜△（a＋b）｜を｜△（a＋b）｜≦｜△a｜＋｜△b｜の関係式からaの誤差｜△a｜とbの誤差｜△b｜の和として絶対値で評価すると、7桁で計算を行なう際に、場合によっては $10^{-7}\times10^{6}=10^{-1}$ となり1分たたない間に誤差だらけになってしまう。実際にはそうはならないことから、誤差に関してはいわゆる「数学的な」センスは役にたたないことがわかる。

これは極端な例ではあるが、実は入門者だけでなくベテランと言われる人も、こういった有限桁の計算によって生じる様々なトラブルに泣かされている。実際、計算量が膨大であるが故に予想もつかない計算結果になることがある。有限桁で計算をしているがためのトラブルなのか、基本的なアルゴリズムが間違っているのか、単なるプログラムのコーディングミスなのか、さらにはそういったことの複合としてトラブルが生じているのか、ほとんど五里霧中的な状態になることも多い。しかもあるパラメータの時はうまくいくのにあるときは駄目などという中途半端なトラブルもあるから困りものである。

従って計算結果についてはその信頼性を実験の時と同様に慎重に確認しなくてはならない。基本的には何回もの計算を、条件を変えながら（計算範囲あるいは、境界条件等を変えるとかしながら）おこない、その結果から判断しておかしいところがあればトライアンドエラーでトラブルの種をつぶしていくということになる。こういったことが数値「実験」と呼ばれる所以であろう。

とはいえ、思いもかけない結果が生じた原因を見つけだすことも数値実験の知的醍醐味といえなくもない（その快感はやったことのある人にしかわからないものである）。この数値実験の実例を、筆者の研究から示すことにしよう。

さらに、数値実験を行なう上でのトラブルの一因となるような、数値計算上の現象について簡単に5節で述べるが、まず数値実験の成果と現状を筆者の専問の分野で説明しよう。

## 4．弾性波の数値実験

### 4．1　弾性波とは

固体中の波を弾性波という。日常的には弾性波という言葉はなじみの薄い言葉だが、地震の波と言えばいちばん分かりやすい。しかし固体中の波は地震だけではない。金属棒を伝わる波も弾性波であるし、超音波といって数十ＫＨｚ（キロヘルツ）ないし数百ＭＨｚ（メガヘルツ）の周波数の固体中の波も弾性波である。

地震の場合に縦揺れと横揺れがあるように、固体中では2種類の速度の波が存在する。これは空気中または水中では一種類の速度の波しか存在しないのと大きく違う点である。従って固体中の波の振舞いは非常に複雑になる。そのため弾性波の研究はコンピュータを用いて解くことが最も早く試みられた分野の一つである。

固体内の波（弾性波）を説明するために、固

体に力が加わったとき固体がどういう変形をするかを考えよう。図1に示すように固体内を等間隔で格子状に区切って、固体の上面に上下方向の正弦波状の力が加わる場合を説明する。

まず下向きの力が加わるとその部分は凹んで縮むが、上向きの力が加わると引っ張られて図1（a）のようになる。これを繰り返すと、力が加わらないときは正方形に区切られていたものが、図1（b）のように、圧縮部と伸長部とが交互に発生して下方に伝播している。これが地震での縦揺れに当たる縦波である。固体中の各格子点の変位を、図2で示すように変位ベクトルu（静止位置での各格子点から変形後の各点に向かうベクトル）で示す。図1の場合、変位ベクトルは上下方向を向く。縦波では進行方向と変位ベクトルの方向が図3のように平行である。

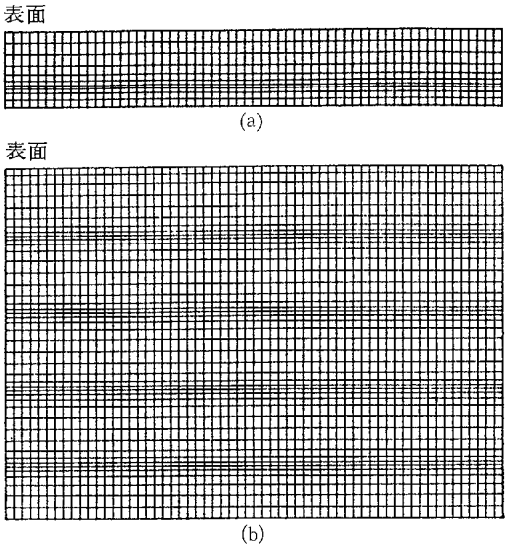


**図1**　縦波の変形の格子図

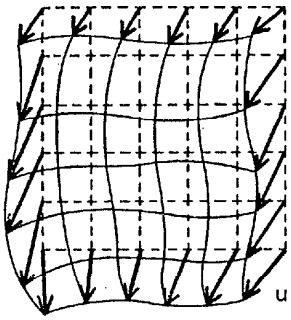


**図2**　変形と変位ベクトル

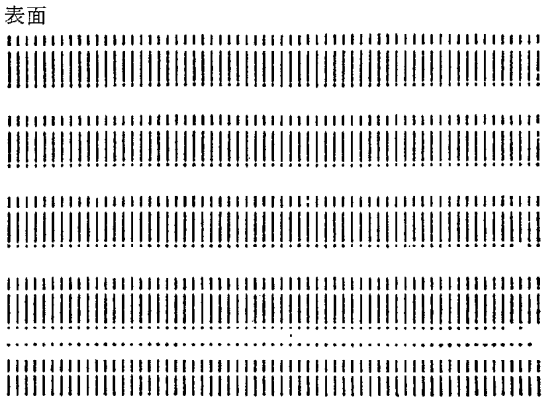


**図3**　縦波の変位ベクトル図

次に固体の表面に平行な力を加えた場合を説明する。まず右方向に力を加えると、正方形だった格子が図4のように菱形の格子に変形する。逆に左方向の力を表面に平行に加えると、逆向きの菱形の変形が生じる。これを繰り返すと図4のような歪変形が下方に伝わる。これが地震の横揺れに当たる。このとき変位ベクトルは図5のように（縦波の時とは逆に）水平方向を向き進行方向に垂直になるので横波と呼ばれる。

これらの縦波、横波の変位ベクトルと格子の変形図を用いて数値実験の結果を表示し、種々の場合の弾性波のメカニズムの解明を4．3で示す。

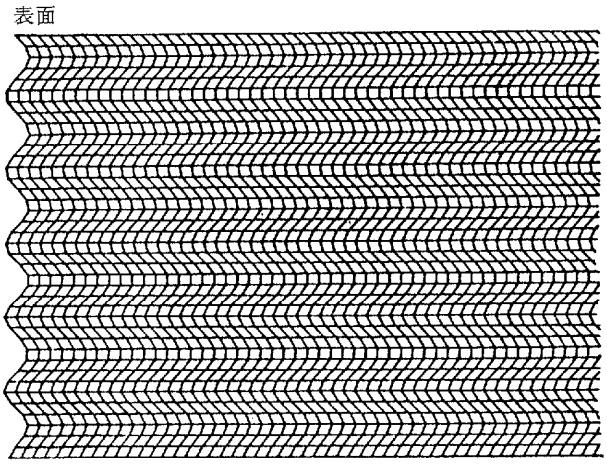


**図4** 横波の変形の格子図

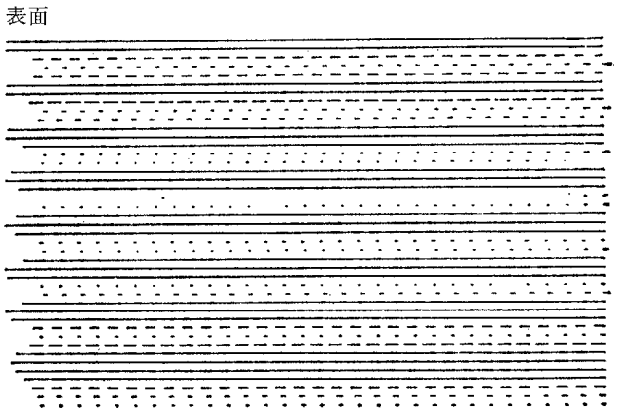


**図5** 横波の変位ベクトル図

## 4．2 弾性波動論の歴史と現状

弾性波動論の基礎は、19世紀にすでに確立されていた。20世紀に入ってからは、多いとはいいがたい研究者たちが地道な研究を行った結果、種々の問題が解かれるようになった。しかし、音波・電波・原子物理などの分野に比べると、その研究の速度はゆっくりとしたものであった。

電波及び音波の大部分の問題や原子物理の問題では、求めるべきものが圧力またはポテンシャンのようなスカラー波動（一点について値が一つ求まればよい）なのに対して、固体内の弾性波は、ベクトル波動である。すなわち、変位が図2で示されるようにベクトルで与えられるため一点についてx方向の変位とy方向の変位の二つの値を（二次元の場合）求める必要がある波ということである。ベクトル波動の基礎方程式は連立偏微分方程式となり、これを解くことが困難であったことが進歩の遅れた主な要因と思われる。

弾性波動論は地震が主な応用分野なので、地震学者を中心とした数少ない理論家が細々と研究を続けていた。例えば20世紀の前半で、音波・電波の反射の論文は数百編あるのに対して、弾性波動では十編以下である。筆者が弾性波の問題にとりかかったのもそのころで残された問題はむずかしい問題ばかりであった。例えば板による弾性波の反射の問題では式を処理し、解析的に解くのに2年かかるといった有様であった。

このように今世紀に入ってからは、与えられている方程式をいかにして解くかということだけが目的になっていた。しかし、一つの問題を解くのに数年を要していたのでは、数多くの応用の目的に答えることは不可能である。これは解析的に弾性波の問題を解くことの限界を示しているものといえよう。ちょうどそのころ、高速大容量（当時としては）の計算機が利用できるようになって、徐々にいろいろな問題が半数値的に解かれるようになってきた（現在のように完全に支配方程式を数値的に解くほどにはハードの性能は高くなかった）。

1960年代に入って、イスラエルの地震学者たちがＣＤＣ社の高速計算機を用いて完全に数値的に解くことを始めた。彼らが角のある問題を解いたと聞いたとき［文献5］、数学的には角は特異点になるので解けないと考えていた筆者は、一晩まんじりともしなかったのを覚えている。それ以来筆者は、解析的方法はやめ数値的にいろいろな問題を解くようになったのである。世界的にみても、静的弾性論では有限要素法（ＦＥＭ）が研究時代を終えて実用時代に入った。同様に動的弾性論でも数値的解法が主流となって実用時代に入ろうとしている。もちろん数値解法にもいろいろな方法があり、解析的に式をたてて問題を解く努力もなされているが、新しい解法は約十年に一度出るか出ないかで、そのうえ新しい解法が従来の方法より優れているとは限らないのが現状である。

## 4．3 弾性波のメカニズム［文献6－8］

本来は、固体中の波を実際に目でみることは不可能なことである。しかし数値実験の結果を図1乃至図5で示されるように、ベクトル表示または格子表示することで固体内弾性波の動きを可視化し、直観的に理解する事が可能になった。

さらに全時間ステップの各メッシュ点の動きを見るためにこれを映画化した。これによって以下のことがわかった。

1）時間的・空間的に拡大することにより各点の動きを空間的には詳細に、時間的にはゆっくりと観察することができる。

2）そのために動的な問題を準静的に考えることができる。

3）実験的に与えることは困難な種々の条件を容易に与えることができる。

4）格子図を用いることによって、2）とともに直観的、物理的に弾性波の動きを理解することができる。

5）従って弾性波の輻射、伝播、反射のメカニズムを理解することができる。

以上のように、差分法または有限要素法等によって超音波のシミュレーションを行なう場合、現象のメカニズムを解明できる点がこの手法の大きな利点であることがわかる。実は、このように数値計算で得られた膨大なデータをグラフィックによって表示するといったことは、他の分野においても最近では盛んに行なわれるようになってきた方法である。

メカニズムの解明の実際を以下に示そう。

### 4．3．1　垂直振動子からの横波発生のメカニズム

固体の表面に振動子（帯状音源）を置き、これに正弦波状の垂直応力を加えると、図1でみたように縦波（圧縮波）が生じることは音波の場合と同様である。さらに弾性波では同時に横波が発生し、斜め前方に伝播する。小さい振動子からの縦波及び横波の遠方での指向性は図6に示されるように、前面ではLで示される縦波が強く、斜め前方にはTで示される横波が強く発生することが理論的にわかっている。またSLで表された部分は、サイドローブと呼ばれる横波の一部で、中心線に近い方向に伝わる。表面に垂直な力を加えているのにどうしてこうした横波が発生するかは従来はわからなかったので、その理由を考えてみよう。

図7にしめされる固体表面に置かれた帯状源に正弦波状の垂直応力が加わると、振動子前面は圧縮、伸長を繰り返して、前面に圧縮波（縦波）Lを発生する。図は中心線に対して対照なので、右半分のみを示している。一方図7の音源の右端付近では、他の部分と非常に異なった変形Sをしていることがわかる。これは音源の前面では圧縮、伸長が繰り返されるのに、音源の外では変形がなため、その中間部の音源の右端附近に強い不規則変形Sを生じるためである。この歪み変形が右下に伝播して、横波Tとなるのである。つまり音源の端に生じた歪み変形Sが横波の原因である。

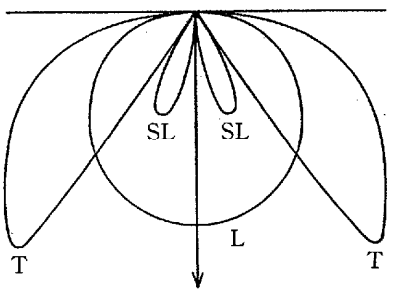


**図6**　小さい振動子の指向性
L：縦波　T：横波　SL：横波のサイドローブ

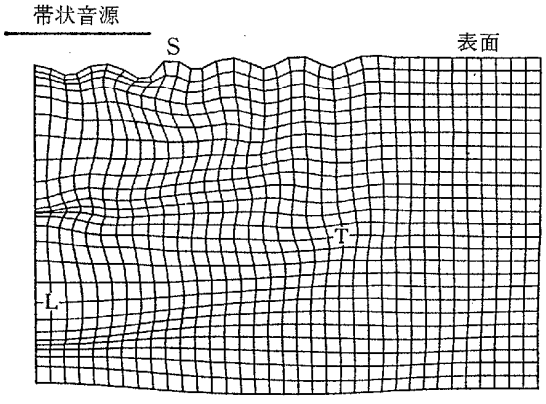


**図7**　帯状音源付近の変形の格子図
L：縦波　T：横波　S：歪み変形

これをベクトル図でみると、図８（ａ）に示されるように、下向きの応力が加わると下向きの変位ベクトルが生じていることがわかる。さらに上向きの応力によって上向きのベクトルが振動子面上で発生し、これによって引き込まれるように端付近にアーチ状のベクトルAが生じる（図８（ｂ））。このアーチ状ベクトルAが図７の歪み変形Sに対応している。ベクトルAの方向（左下向き）は進行方向（右下）と垂直なので、横波Tとして右下方向に伝播する。これが図６の指向性図のTになる。

また表面（図９の最上部）に沿って右方向に伝わっている波を表面波という。これは表面にのみ限定された波で、図９の格子図でSで示された水の表面波に似た波であって、図10のベクトル図では表面付近の上下向きのベクトルで示されるSに対応している。従来の理論ではこの表面波も図11に示されるようにｘ方向の変位ｕとｙ方向の変位ｖの垂直方向の分布として与えられていたので表面波の物理的な意味（メカニズム）がわかりにくかったがシミュレーションによってその理解が容易になった。さらに、映画化することによって、上記のような数枚の図で結果を示す以上に、これら音源付近の固体の運動を詳細に見ることができる。

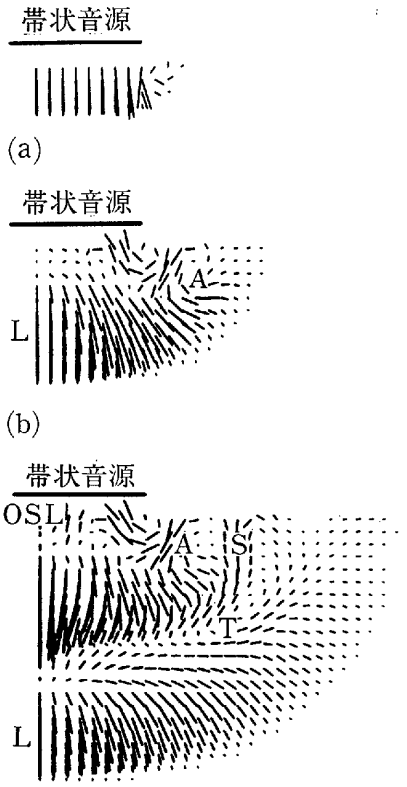


**図８**　帯状音源付近の変位ベクトル図
Ｌ：縦波　Ｔ：横波　Ａ：アーチ状ベクトル
ＯＳＬ：横波のサイドローブの起源ベクトル

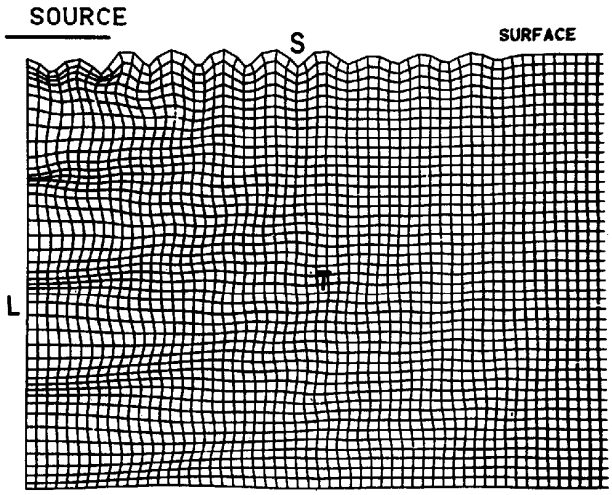


**図９**　表面波及び垂直振動子より
輻射の格子図

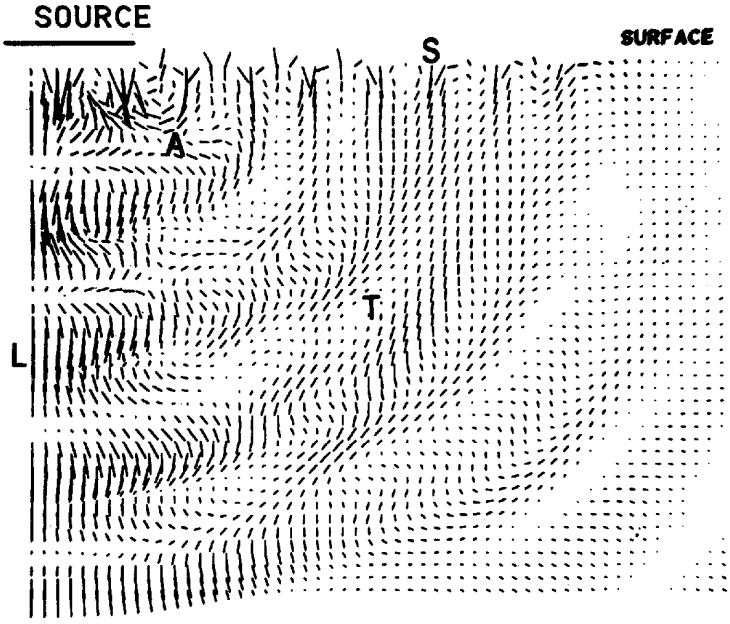


**図10** 表面波及び垂直振動子より
輻射のベクトル図

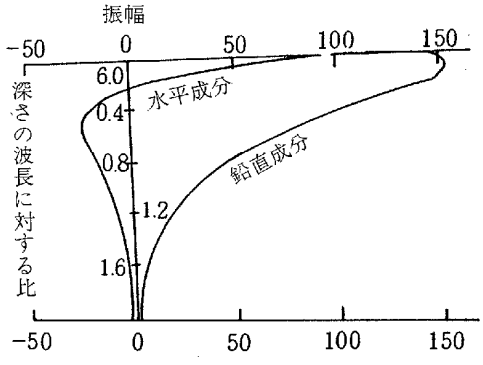


**図11** Rayleigh 波による運動の表面及び内部における変位の振幅（Milne, Lee による）

4．3．2 地震波のメカニズム

われわれに身近な地震波についても、従来より理論的に様々なことが解明されてきた。これらの研究は難解な数学を用いざるを得なかったために、研究の進歩も遅々としたものであった。研究者の数も、我国のような地震国においてさえ、理論家と言えるような人は多くても10人を越えることはないような有様だった。また、得られた結果にしても、理解するのに多大の努力がいるようなものが多かった。

現在、ここでもシミュレーションによって容易に現象が理解できる状況になってきている。

例）二層地盤での各種モードの物理的メカニズム

1層の板中を伝わる弾性波は、板波として各種のモードが存在し、それぞれのモードが中心線に関して対称であるか、非対称であるかなどの物理的な意味もよく知られている。2層の場合も、より複雑ではあるが、同様に各種のモードが存在する。

図12に表面波が二層の地盤上を伝播するときの基本モード、第一モードの変位の大きさの、深さによる変化を理論によって求めたものを示す。図の最上部は表面で、下方の水平線は二層の境界を表す。実線は水平変位の値、破線は垂直変位の値のそれぞれ理論値である。黒点及び中空円は数値実験による値で、理論値とよく一致している。右側の基本モードは表面では垂直成分が大きく、水平成分は表面に近いところで正負が逆になるが、全体とした垂直成分が水平成分より大きく、図11に示される通常の表面波のそれとよく似ていることがわかる。一方、左の第一モードは表面波のそれとは大きく異なり、全体としては水平成分が垂直成分より大きいことがわかる。

これをシミュレーションの結果のベクトル図で示したのが図13である。図の右側から伝播してきた表面波が二層地盤に到達した後、下層の地盤の方が伝播速度が大きいため、上方地盤中の表面波（R0）より先行した下方地盤の境界面波（R1）が見られる。R0が基本モードであり、R1が第一モードである。R0は通常の表面波と同様に垂直成分が大きいことが図からわかる。一方、R1は下方の地盤中では通常の表面波と同様に垂直成分が大きいことが図13で見られるが、理論値の方も、図12からわかるように、深さが30m以上の地盤中で垂直成分が大きいことを示している。しかし、深さが30m以下の上層中では水平成分が大きくなっている。

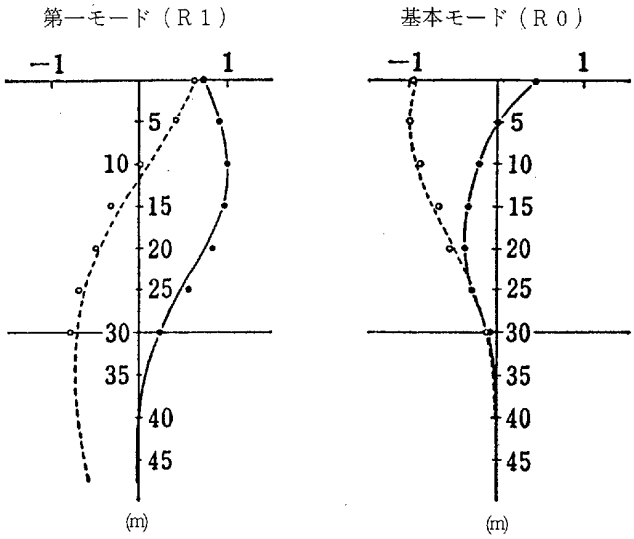


**図12** 二層地盤上での表面波の基本モード（R0）と第一モード（R1）の変位の垂直分布（理論値）
実線：水平成分　破線：垂直成分　・◦：数値実験値

これを図13のベクトル図でみると、下層の地盤中で表面波の前面（左側）と後面（右側）は上向きの垂直成分が大きいことがわかる。これが上層部にはいると、後面（右側）から前面（左側）に押されたかのように左向きの水平成分を持つベクトルを生じる。従って上層では水平成分が垂直成分より大きくなって、通常の表面波と異なった成分になっているのである。

図12の理論値だけではＲ０、Ｒ１の各成分の大きさはわかっても、これらのモードの物理的意味がとらえにくいが、シミュレーションの結果をベクトル図で表すと、その意味が容易に理解できることがわかる。

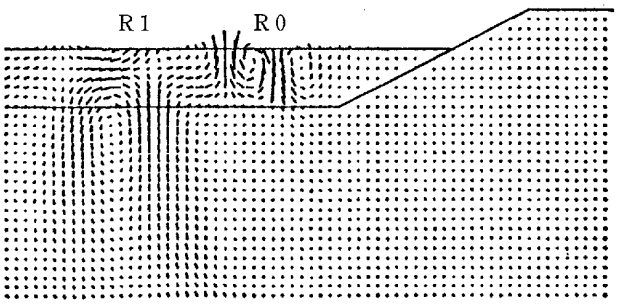


**図13** 二層地盤上を左方に伝播する表面波の変位ベクトル図
Ｒ０：基本モード　　Ｒ１：第一モード

## ４．４　弾性波のシミュレーションの応用

固体内の傷を数MHz（$10^6$ ヘルツ）の周波数の超音波を用いて発見する超音波探傷という技術がある。従来は傷があるかないかだけでもわかれば良いと考えられていた。しかし近年、傷の大きさをも測定する必要が生じ、現在では測定技術の向上により傷の大きさの定量化が行なわれるようになってきている。これはシミュレーションの結果の一つの応用といえるので、簡単に説明しよう

図14はガラス内の超音波をストロボ光を用いて可視化した写真である。これに対応するシミュレーションのベクトル図が図15である。シミュレーションの結果と光弾性可視化の結果がよく一致しているのがわかる。入射波ＩＴが先端が方形の傷によって反射されている。右側の面による反射波がＲＴである。ＲＴ１、ＲＴ２は入射方向に帰ってくる、上端の二つの角を中心とした反射横波で、ＲＴ３は反射方向に進行する、傷の上面で反射された横波である。これらのＲＴ１、ＲＴ２、ＲＴ３は傷の上端を中心とした波になっているためこれらを利用すれば傷の上端の位置がわかる。同様に下端を中心とした反射波も発生するから両端の位置が測定できる。さらに、ＲＴ１、ＲＴ２の方向の二つの波とＲＴ３の方向の一つの波を実験で確かめた写真を図16(a)、(b)に示す。

このようにシミュレーションで予測された現象が、実験で確認されることが現在では行なわれるようになってきた。すなわち、従来現象を発見するための道具であった実験と同様の機能を、コンピュータシミュレーションが果たすようになってきたということである。

このように、新たな現象がシミュレーションによって発見され、その結果を用いて実際の測定が改良されるようになってきた。さきに述べた弾性波のメカニズムの解明が理学的応用であるとすると、これは工学的な応用であるといえる。

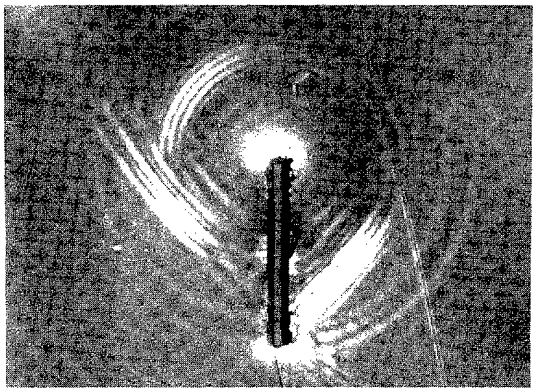

**図14** 欠陥による横波の反射の光弾性図

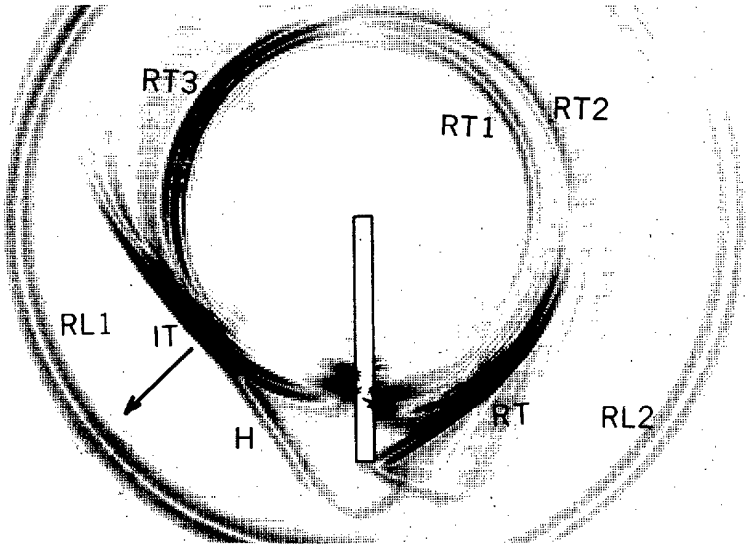


**図15** 図14に対応する変位ベクトル図

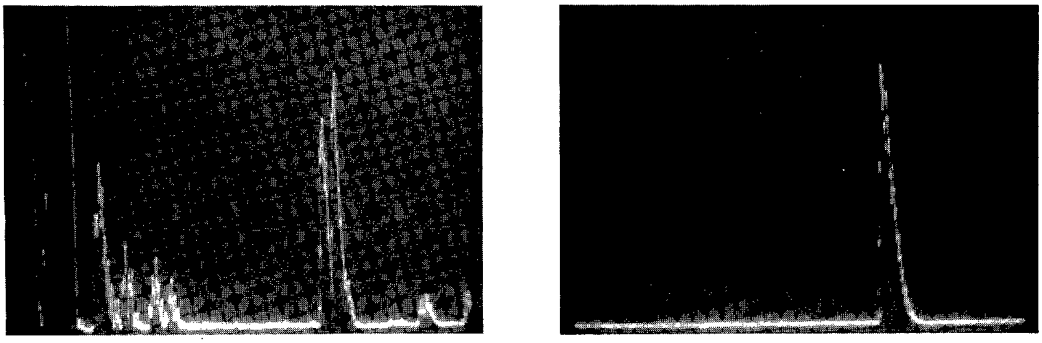

**図16** (a) 図15のRT 1、RT 2の双峰性のパルス　(b) RT 3の単峰性のパルス

## 5．数値計算における落し穴

コンピュータで計算をする際に初心者が犯しやすい失敗は、既製のプログラム（例えば有限要素法による構造解析プログラム等）にデータをいれ、計算機から出て来た結果を正しいかどうかをチェックもせずに無条件に信用してしまうことである。

コンピュータは間違わないとしても（もっともハードウェアにしても、人間の作ったものであるから100％信頼できるというものでもないが）ソフトウェアの方はそういうわけにはいかない（基本ソフトであるオペレーティングシステム（ＯＳ）にもバグ（間違い）はごろごろあるそうだ）。人間が欠陥だらけで、間違いをしない方が不思議なくらいだという実感が、コンピュータを使っているといやでもわいてくる。十分にデバッグをしてあるはずのソフトでも、ソフトの規模が大きいとどうしてもバグがなくならないといったことになる。前述したＯＳがいい例である。ソフトの信頼性の評価はソフトが大きければ大きいほど難しい問題となってくる［文献９］。

数値計算用ソフトウエアでは$10^{-50} < x < 10^{+50}$迄の広い範囲の数値についてバグがあってはいけない。しかし実際にはこの検証は非常に困難であるので、実際問題としては困難さの認識と、慎重な利用が必要である。[文献10]

ジャンボ機の設計に従事し、数値計算には長年の経験を持つ、ボーイング社のもと計算部長のコルビン氏も、部下が計算結果をよくチェックしないで間違った結果を無条件に信用することがいちばん危険だというコメントを述べていた。これを防ぐには慎重すぎるくらいの慎重さが必要である。さらに、どういう場合に計算ミスが生じるか、といったことも知っていなくてはならない。計算機の出力を信用し過ぎるなということを肝に命じながら計算機を利用する必要があろう。特に数値計算をする場合、今まで親しんできた？数学の感覚が抜けないで失敗することは良く見かけられる。[文献11、12]このことについては簡単に既述したが、もう少し詳しく２、３の例を紹介しよう。

例１．0.1をＮ回加えるとどうなるか

大部分のコンピュータの内部では数値は２進数で表されていることはほとんどの人が知っていると思うが、実際に利用しているときにそのことを気にしながら利用しているユーザーはほんの一握りであろう（たいていの場合は気にする必要が無いし、又ユーザーフレンドリーという観点にたてば気にしなくてはいけないようなシステムでは困るのではあるが）。数値計算を始めたばかりの学生などがいちばん冒し易いミスは、0.1とか0.01とかを100回加えたとき10.0とか1.0になると信じていることであろう。

```
10 A = .1
20 B = 0!
30 FOR K = 1 TO 100
40 FOR I = 1 TO 10
50 B = B + A
60 NEXT I
65 B# = B
70 LPRINT USING"####.##########";B#
75 NEXT K
80 STOP
90 END
```

**図17** 0.1（10進）を10回加えたごとに値をプリントするプログラム

IF X＝10.0THEN……などとして、Xが10.0になったらループの外に出るようなプログラムをつくって、いつまでたってもループの外に出ないと頭を抱えているのをよく見かける。

実際にパソコンで0.1を次々と加算して行くとどうなるかを試してみよう。プログラムは図17に示したような簡単なもので、BASICで書かれている。加算の結果は10回ごとに出力するようになっている。ここではBASICでの結果のみを示すが、興味を持たれたならば、C、Pascal、Fortran等他の言語で試みられたい。

0.1をN回加えた結果を小数点以下9桁まで出したのが表1である。N＝3までは下2桁が01、03、12となっていることから、誤差は増加していることが解る。N＝5までは減少し、N＝19までは誤差が再び増加し、N＝80までは減少していることがわかる。

以前、ミニコンで（ハード的にみれば今のパソコン以下かも知れない）同様の計算をした結果を表2に示す［文献12、13］。この場合、N＝40までの結果はマイナスの誤差が増加、160まではプラスの方に変わり、640までは再び真値より小さな値をとっている。パソコン（表1）とミニコン（表2）とを、N＝2560で比較すると、パソコンは256.0064でミニコンの0.2560109E03＝256.0109よりわずかに誤差が小さいことがわかる。

| N | 実測値 | N | 実測値 |
|---|---|---|---|
| 1 | 0.100000001 | 21 | 2.100000143 |
| 2 | 0.200000003 | 22 | 2.200000048 |
| 3 | 0.300000012 | 23 | 2.299999952 |
| 4 | 0.400000006 | 78 | 7.799994946 |
| 5 | 0.500000000 | 79 | 7.899994850 |
| 6 | 0.600000024 | 80 | 7.999994755 |
| 7 | 0.700000048 | 81 | 8.099994659 |
| 8 | 0.800000072 | 82 | 8.199995041 |
| 9 | 0.900000095 | 83 | 8.299995422 |
| 10 | 1.000000119 | 84 | 8.399995804 |
| 11 | 1.100000143 | 85 | 8.499996185 |
| 12 | 1.200000167 | 240 | 24.000055313 |
| 13 | 1.300000191 | 280 | 28.000070572 |
| 14 | 1.400000215 | 320 | 32.000083923 |
| 15 | 1.500000238 | 360 | 36.000022888 |
| 16 | 1.600000262 | 400 | 39.999961853 |
| 17 | 1.700000286 | 2520 | 252.006179810 |
| 18 | 1.800000310 | 2560 | 256.006408691 |
| 19 | 1.900000334 | 2600 | 260.006652832 |
| 20 | 2.000000238 | | |

**表1** パソコンで0.1（10進）を$N$回加えた実測値

| N | 実測値 |
|---|---|
| 40 | 0.3999997E＋01 |
| 160 | 0.1600004E＋02 |
| 640 | 0.6399933E＋02 |
| 2560 | 0.2560109E＋03 |

**表2** ミニコンで0.1を$N$回加えた実測値

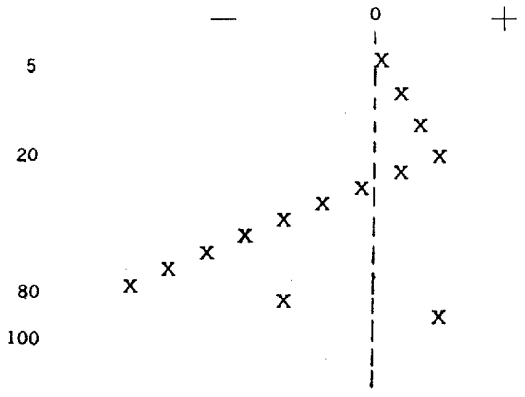


**図18** パソコンにおける丸め誤差の変化
(0.1をN回加えたとき)

```
10 A =0.1
20 AD = VARPTR(A)
30 FOR I = 0 TO 3
40 LPRINT HEX$(PEEK(AD+3-I))
50 NEXT
60 END
```

**図19** 0.1を16進表示するプログラム

図18に、パソコンの場合の誤差をグラフにして示した。誤差が一定の割合で増減している様子がわかる。この理由を調べるために0.1の16進表現を図19のプログラムで求めてみると7D4 CCCCDとなる。7Dが8ビットの指数部を、4 CCCCDが24ビットの仮数部を表している。このパソコンでは指数部は80(16進)が0でそれ以下は負の値であるから、7D=「0111 1101」(2進)、80=「1000 0000」(2進)、80−7D=「0000 0011」=3より7Dは−3を表していることになり、$2^{-3}$を意味している。仮数部の4CCCCDは、2進表示してみると「0100 1100 1100 1100 1100 1101」であるが、4=「0100」の頭は符号ビットで、仮数表示の1ビットが隠されている。つまり実際の仮数部は、常に先頭ビット1を入れなければならないから1100となる。従って固定2進表現で表すと、

$$0.1 = .0001\ 1001\ 1001\ 1001\ 1001\ 1001\ 101 \qquad (1)$$

となる。次に0.2+0.1は固定2進では、

$$\begin{array}{l} 0.2 = .0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 01^{\triangle} \\ +0.1 = .0001\ 1001\ 1001\ 1001\ 1001\ 1001\ 101 \\ \hline 0.3 = .0100\ 1100\ 1100\ 1100\ 1100\ 1100\ 111 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 01\ 00 \end{array} \qquad (2)$$

となる。0.2の25桁目(△の桁)は“1”であるので0捨1入をすると切り上がり、0.3の下4桁は「0100」となる。又0.4+0.1は、

$$\begin{array}{l} 0.4 = .0110\ 0110\ 0110\ 0110\ 0110\ 0110\ 1^{\triangle} \\ +0.1 = .0001\ 1001\ 1001\ 1001\ 1001\ 1001\ 101 \\ \hline 0.5 = .1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 001 \end{array} \qquad (3)$$

と0.5になるが、25桁目(△の桁)が“0”なので0捨1入して切り捨てられる。又0.5+0.1は、

$$\begin{array}{l} 0.5 = .1000\ 0000\ 0000\ 0000\ 0000\ 0000^{\triangle} \\ +0.1 = .0001\ 1001\ 1001\ 1001\ 1001\ 1001\ 101 \\ \hline 0.6 = .1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 101 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 1010 \end{array} \qquad (4)$$

となるが、25桁目が“1”なので切り上がって0.6の下4桁が「1010」となる。

以上より、0.1を加算したときの切上げか切捨てかは25桁目が0か1かで決まるが0.2に対しては0.1の24桁目の“1”、0.4に対しては23桁目の“0”、0.5に対しては22桁目の“1”が、それぞれ25桁目に来る。

式(1)の0.1の2進数の下10桁とそれに対するNの値は式(5)のようになる。例えば、0.4=0.1

×4でN＝4となる。

| N | 640 | 320 | 160 | 80 | 40 | 20 | 10 | 5 | 4 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

(5)

0.1×N＋0.1でN＝2では“1”で切上げ、0.4のN＝4では“0”で切り捨て、N＝5～19までは“1”で切上げ、N＝20～79までは“0”で切捨てとなり、表1及び図18のような誤差の増減が生じる。

ミニコンでは仮数部は23ビットなので2進表示の下11桁は、

| N | 640 | 320 | 160 | 80 | 40 | 20 | 10 | 5 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 値 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

(6)

となり、N＝5～19は切捨て、N＝20～79は切上げと、パソコンの場合と違って表2のようになる。パソコンの方が誤差が小さいことがわかる。

又パソコンで、N＝20～79は切捨てになって0.1の下4桁「1101」が切り捨てられ、「1101」$=2^{-24}\times1.625\fallingdotseq96\times10^{-9}$であるから、表1で2.0～7.9までほぼこの値の割合で減少していることがわかる。誤差εをN回演算すると、全体の誤差は$\varepsilon/\sqrt{N}$になると数学では教えるが、これは誤差が正規分布の場合であって計算機ではそうはならないことに注意して欲しい。

例2　桁落ち

（EXP(－X)と1／EXP(X)と　は違う）

数値計算の誤差に大きく影響するのは上に述べた丸め誤差よりも、ここで述べる桁落ちである。しかし、桁落ちがしばしば起きて、計算結果に重大な影響を及ぼすことは案外知られていない。

すでに知っている人も多いと思うが、桁落ちとは絶対値が同じくらいの数の加減算によって有効桁がなくなることである。例えば、

1.23456－1.23450＝0.00006

この場合、有効桁6桁の数が有効桁1桁の数になってしまい、5桁有効桁が失われている。このとき5桁の桁落ちをしたという。こうなると1桁の精度しかないため、これ以降この数値を用いた計算をすれば、その結果の精度は低下する。

しかるにパソコンのBASIC（MS－BASIC）のマニュアルの数学関数の項を見て唖然となった。というのも、そこには

$$\sinh(x)=(\exp(x)-\exp(-x))/2 \qquad (7)$$

を用いろと書かれていたからである。この式はx＝0で桁落ちを生じる例なのである。つまりこの式はx＝0の近くでは使用できないのである。ここではこうした桁落ちについて見ていくことにしよう。

sinh(x)の値（x＝0.001から0.01まで）を

$$SH=(EXP(x)-EXP(-X))/2.0 \qquad (8)$$

$$STH=(EXP(x)-1.0/EXP(x))/2.0 \qquad (9)$$

の2通りの方法で2社のパソコンを用いて計算した値を真値とともに表3に示す。表から、このような小さいxの値ではexp（x）を用いてsinh（x）を求めることは危険であることが理解できよう。又計算方法の違いにより結果が違ってくることも、とくにx＝0.005でSH＝0.004994..と正解の0.00500..と違うことに気がつかれよう。参考までに、表4にEXP(x)、EXP(－x)、1/EXP(x)を示す。

表3でB社の場合は0.01付近でも2桁落ちし

| $x$ | A 社のパソコン | | B 社のパソコン | | sinh の真値 |
|---|---|---|---|---|---|
| | SH | STH | SH | STH | |
| 0.001 | 0.001000017 | 0.001000017 | 0.0010000200 | 0.0010000200 | $1.000000167\times10^{-3}$ |
| 0.002 | 0.002000004 | 0.002000004 | 0.0020000000 | 0.0020000000 | $2.000001335\times10^{-3}$ |
| 0.003 | 0.003000021 | 0.003000051 | 0.0030000200 | 0.0030000500 | $3.0000045\times10^{-3}$ |
| 0.004 | 0.004000008 | 0.004000068 | 0.0040000400 | 0.0040000700 | $4.00001067\times10^{-3}$ |
| 0.005 | 0.004999995 | 0.005000025 | 0.0050000300 | 0.0050000300 | $5.000020775\times10^{-3}$ |
| 0.006 | 0.006000012 | 0.006000042 | 0.0060000400 | 0.0060000400 | $6.00003595\times10^{-3}$ |
| 0.007 | 0.007000029 | 0.007000029 | 0.0070000900 | 0.0070001500 | $7.000057105\times10^{-3}$ |
| 0.008 | 0.008000076 | 0.008000076 | 0.0080001400 | 0.0080001900 | $8.00008533\times10^{-3}$ |
| 0.009 | 0.009000093 | 0.009000093 | 0.0090001800 | 0.0090002100 | $9.00012143\times10^{-3}$ |
| 0.010 | 0.010000169 | 0.010000169 | 0.0100002000 | 0.0100002000 | $0.010000166\times10^{0}$ |

$$SH=\frac{e^{x}-e^{-x}}{2}\qquad STH=\left(e^{x}-\frac{1}{e^{x}}\right)/2$$

**表3** sinh $x$の値を二つの方式で計算する

| $x$ | EXP ($x$) | EXP ($-x$) | 1/EXP ($x$) | exp($-x$)の真値 |
|---|---|---|---|---|
| 0.001 | 1.001000524 | 0.999000490 | 0.999000490 | 0.999000499 |
| 0.002 | 1.002002001 | 0.998001993 | 0.998001993 | 0.998001998 |
| 0.003 | 1.003004551 | 0.997004509 | 0.997004449 | 0.997004495 |
| 0.004 | 1.004008055 | 0.996008039 | 0.996007919 | 0.996007989 |
| 0.005 | 1.005012512 | 0.995012522 | 0.995012462 | 0.995012479 |
| 0.006 | 1.006018043 | 0.994018018 | 0.994017959 | 0.994017964 |
| 0.007 | 1.007024527 | 0.993024468 | 0.993024468 | 0.993024442 |
| 0.008 | 1.008032084 | 0.992031932 | 0.992031932 | 0.992031914 |
| 0.009 | 1.009040594 | 0.991040409 | 0.991040409 | 0.991040378 |
| 0.010 | 1.010050178 | 0.990049839 | 0.990049839 | 0.990049833 |

**表4** exp($-x$)と1/exp($x$)とは値が異なる

6桁の精度になっていることがわかる。又0.002付近では3桁落ちして5桁の精度になっている。また表3からもわかるように、同じ方法でもパソコンの機種が異なると結果も違っている。

x＞1/2ではexp（x）－exp（－x）は桁落ちしないがx＜1/2のような小さい値に対しては桁落ちするので級数展開を用いるべきである。もっともsinh（x）の値がこんなに小さい値になるのだから4桁も合えば十分だという人もいるかも知れない。そういう人のためにcoth（x）の値を

$$COTH=(\exp(x)+\exp(-x))/(\exp(x)-\exp(-x)) \tag{10}$$

$$COTT=(\exp(x)+1/\exp(x))/(\exp(x)-1/\exp(x)) \tag{11}$$

$$COTW=(\exp(2x)+1)/(\exp(2x)-1) \tag{12}$$

の3種の方法で計算した結果を表5に示す。このようにsinh（x）で割ると結果は大きい数になるので小さい数は精度が悪くても良いというのは暴論で、時と場合によっては思いもかけない悪影響を及ぼすこともある。

| $x$ | COTH | COTT | COTW | coth の真値 |
|---|---|---|---|---|
| 0.001 | 999.98352051 | 999.98352051 | 1000.00061035 | 1000.000339 |
| 0.002 | 500.00000000 | 500.00000000 | 499.99517822 | 499.9993347 |
| 0.003 | 333.33251953 | 333.32919312 | 333.33398438 | 333.3343331 |
| 0.004 | 250.00152588 | 249.99777222 | 250.00138855 | 250.0013337 |
| 0.005 | 200.00270081 | 200.00149536 | 200.00146484 | 200.0016668 |
| 0.006 | 166.66932678 | 166.66850281 | 166.66816711 | 166.6686666 |
| 0.007 | 142.86006165 | 142.86006165 | 142.86001587 | 142.8594762 |
| 0.008 | 125.00282288 | 125.00282288 | 125.00287628 | 125.0026667 |
| 0.009 | 111.11447144 | 111.11447144 | 111.11417389 | 111.1141111 |
| 0.010 | 100.00331116 | 100.00331116 | 100.00341034 | 100.0033333 |

**表5** coth $x$の値を三つの方式で計算する

表5で真値と比較すると、xが0.001から0.005までは通常用いられるCOTW＝（exp（2x）＋1）/（exp（2x）－1））の方法が他のふたつの方法より良い値であるのに、0.01では他のふたつの方法のほうがCOTWより真値に近い値をとっており、どの方法が良いのかは、やってみないとわからないという一面を見せている。

もう一つ桁落ちの例として、次のチェビシェフ多項式

$$T_{12}(x)=2048x^{12}-6144x^{10}+6912x^{8}-3584x^{6}+840x^{4}-72x^{2}+1$$

は、$|x|\leqq 1.0$以下で1.0以下の値となり3桁～4桁の桁落ちをする。こういうことを知らずにプログラムを式通りに書いて計算結果を信用すると、とんでもないことになる。以前筆者も、より高次のチェビシェフ多項式を使ったところ、予期しない答えが出て原因解明に数日を要した経験がある。そのときは4倍長計算によってやっと正解を得た。

桁落ちを生じる場合の根本解決は、必要桁数に桁落ちした桁数を加えた桁数で計算を行なうことである。しかし問題によっては問題の設定自体が悪く、数値計算上は正しい答えが出ても、本当に意味のある解かどうかは疑わしいこともある。

例3　意味のない連立方程式の解

例えば連立一次方程式で独立でない二つの方程式を作ったような場合に、意味のない解を得ることがある。

次の連立一次方程式

$$5x+7y+9z=21$$
$$4x+5y+6z=15$$
$$x+2y+3z=6$$

の解は、$x=z$、$y=3-2z$で独立な解とはいえない。ガウスの消去法で丸め誤差なしで解くと0除算を生じる。しかし係数にほんの少しの誤差が入っているとそれに応じて適当な答えが出てくる。例えば係数の9桁目または10桁目に1を加えると9桁の桁落ちを生ずるが、答えは出てくる。たとえば表6のように10桁目に1を入れた場合表6下の$x=1.0645$、$y=0.8709$のような解が得られる。計算の8桁目を1だけ変えると7桁桁落ちして表7のような解が得られる。

データの10桁目、または8桁目は6桁出力すると表示されないのでデータを変えたことはわからない。データを6桁だけ表示して、10桁目を変えて任意の解を出すことができるのは、x、y、zが互いに独立ではないからである。

このようにもともとが独立でないものを変数にとるということは問題の設定が悪いと言うことで、数値計算上で桁数を増やして解けたとしても意味はない。

| | $x$ の係数 | $y$ の係数 | $z$ の係数 | 定数 |
|---|---|---|---|---|
| 第1式 | 5 | 7 | 9.000000001 | 21 |
| 第2式 | 4 | 5.000000001 | 6 | 15 |
| 第3式 | 1 | 2.0000000001 | 3.0000000001 | 6 |

**表6** 式(21)の係数に小数点下9桁（10桁）目に1を入れてガウスの消去法で解く

$x=$ 1.0645200
$y=$ 0.8709680
$z=$ 1.0645200

| | $x$ の係数 | $y$ の係数 | $z$ の係数 | 定数 |
|---|---|---|---|---|
| 第1式 | 5 | 7 | 9 | 21 |
| 第2式 | 4 | 5.0000001 | 6.0000001 | 15 |
| 第3式 | 1 | 2.00000001 | 3.00000001 | 6. |

**表7** 式(21)の係数に小数点下7桁（8桁）目に1を入れてガウスの消去法で解く

$x=$ 3.0000000
$y=$ -3.0000000
$z=$ 3.0000000

リニア・プログラミング（ＬＰ）では、パラメータの数を増やすと正確な解が得られると考えて、多くのパラメータを採ったために、独立でないパラメータをとってしまって、大失敗をすることがよくあるので注意をしなければならない。

例4　多項式近似

パラメータの数が多い方が正確に近いとは言えないも一つの例を次に示そう。［文献14］

$y=1/(1+x^4)$ を(a) $x=0$，1，2，3，4，5の6点において一致する5次の多項式と(b) $x=-5$，$-4$，$-3$，$-2$，$-1$，0，1，2，3，4，5の11点において一致する10次の多項式で近似した場合、(a)より(b)の10次式の方が近似がよいと考えがちであるが、図20の結果を見れば(ｂ）の点線は $x=5$ の付近で実線 $y=1/(1+x^4)$ と食い違っていることがわかる。これは多項式近似が小さい区間のみ良い近似を与えるという簡単なことを忘れたために起こった失敗である。現在では多項式（ラグランジュ）近似のプログラムにデータをいれれば近似式はただちにコンピュータから出てくる。それをグラフにしてチェックすることを忘れると、こうした失敗をする。

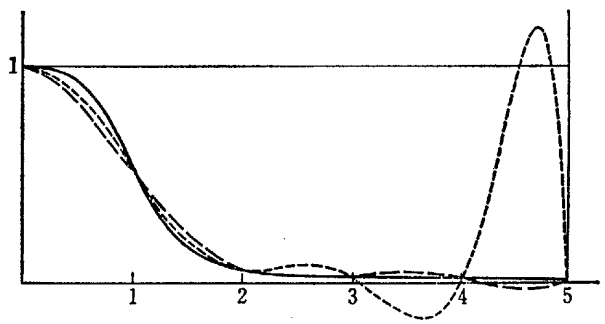


**図20** $\frac{1}{1\times x^4}$ とラグランジュ補間による値との比較〔(ａ)破線：５次のラグランジュ補間の値、(ｂ)点線：10次のラグランジュ補間の値、(ｃ)実線：真値〕

## 6．おわりに

コンピュータの科学技術計算面の現状と問題点のごく一部を紹介した。コンピュータの性能の向上と価格の低下には目を見張らせるものがある。特に最近のワークステーションの分野では、年に一つか二つ画期的な製品が発表されている。例えば現在では10～60 MFLOPS（毎秒数千万回の浮動小数点演算、これは数年前の十億円程度のスーパーコンピュータCRAY 1の1/3～1/2の計算速度）のワークステーションが千万～二千万の価格で数社により発表、または、販売され始めている。

これは、従来日本国内で数十台しかなかったスーパーコンピュータが各工場・各部局に普及することが可能になったことを意味しており、現にそうした傾向が出てきているようである。こうしたワークステーションは主としてシミュレーションに関連した目的で（数値計算そのもの、またはその結果の高速なグラフィック化等に）用いられることであろう。楽観的なことを述べてきたが、ここで思い出すのは、ミニコンまたはパソコンの導入がはやってきたころ、各会社で競って導入したものの、使いこなす人がいなくてこれらのコンピュータは倉庫の中に放置されていたという話である。

コンピュータシミュレーションの普及にはバグの少ない使い易いソフトウエアが不可欠で、これからのコンピュータには一層の使いやすさというものが望まれよう。

コンピュータシミュレーションは理論と実験との中間に位置し、確固たる一分野となるほど大きい意味を持ってきたと信じている。なるほど、ソフトウェアーの開発には人手と時間がかかり、完全なソフトウェアーを作り上げることは決して簡単ではない。しかし、理論では余りに無力であったという面もあるし、実験では困難、あるいは不可能なこともコンピュータシミュレーションで可能になっている。

ここまで、数値実験の持つ重要性と意外性の一端についてを述べてきたつもりである。しかし余りにも広い分野でもあるので、興味をもたれた方は参考文献によって重ねて勉強していただければ幸いである。現在では大学の数学教育も、広くはコンピュータ、狭くはシミュレーションの出現によって大きく変わりつつある［文献4］。

私は大学の目的の一つは「知的なおもしろさを知ってもらう」ことだと考えている。知的な興味を満足させることは、本人の意志と重要性の理解さえあれば、他の欲望の追求よりは容易ではないかと思う。

幸いにしてコンピュータは知的興味を発見するのに非常によい道具である。数値実験はパソコンでも十分実現できることはすでに述べてあるが、注意深さと根気があれば、自ら新しい発見（たとえ些細なものでも）をして知的な喜びを味わうことは、比較的容易なことではなかろうか。

本学はコンピュータの利用あるいはコンピュータそのものを主たる目的としている。従って知的な興味の充足と仕事とが一致し得る環境にあるといえる。

これからは、誰もがコンピュータを使える時代に、使わざるを得ない時代になるであろう。そのとき我々に求められるのは、単にコンピュータを使えるというだけでは不十分であろう。

二三の例で説明したように、情報量が多いほうが良いとは言えない。情報と言っても色々あるが、情報の中には不要どころか有害な情報もあるから、すべての問題に注意深く、多量の情報の中から我々にとって真に必要な情報とは何かを間違いなく判断できる慎重さと叡知とが必要ではなかろうか。我々の思考ではこうした情報の取捨選択をしているようだから。

**参考文献**


1．数理科学“計算理学特集”，No. 268，10月，サイエンス社(1985)
2．コンピュートロール“計算力学特集”，No. 8，10月，コロナ社（1984）
3．“コンピュータの現在”数学セミナー別冊「コンピュータと数学」，No. 1，日本評論社（1985）
4．“数学研究へのコンピュータの影響”数学セミナー別冊「コンピュータと数学」，No. 4，日本評論社（1986）
5．Z. S. Alterman, A. Rotenberg, “Seismic Waves in a Quarter Plane”, Bull. Seism. Soc. Amer., 59, p. 347（1969）
6．K. Harumi et al., “Mechanism of the Generation of Transverse Waves and Suppression of Transverse Waves”, Proc. 9th WCNDT in Melbourne, 4 H10（1979）
7．K. Harumi et al., “Educational Film for Ultrasonic Engineers”, Proc. IEEE Ultrasonics Symp., p. 1074（1982）
8．K. Harumi, “Computer Simulation of Ultrasonics in a Solid”, NDT International 19, p. 315（1986）
9．石井康雄編“ソフトウエアの検査と品質保証”，日科技連出版社（1986）
10. W.J.Cody : Software for Elementary Functions, in “Mathematical Software” p. 171－186 Academic Press.
11. G. E. Forsyth, “Pitfalls in Computation, or why a Mathematical Book Is'nt Enough. “Tech. Report No. Cs 147, Computer Science Deparment, Stanford University（1970）
12. 春海，檜山，小竹，「ミニコンによる数値計算の落し穴」，“bit”，3月～9月，共立出版（1976）
13. “数値演算プロセッサー”，インターフェース別冊，CQ出版（1987）
14. 高田，春海編，“数値計算の手順と実際”，コロナ社（1984）
15. 宇野利雄，“計算機のための数値計算”，浅倉書店（1963）
16. 伊理正夫，藤野和建，“数値計算の常識”，共立出版（1985）