科学技術計算システムの動向

日立製作所　情報システム工場
小国　力

1．物理現象からプログラムまで

2．これまでの科学技術計算ソフトウェアの歩み

3．最近のコンピュータ技術の発展

4．これからの科学技術計算システム

1．物理現象からプログラムまで

コンピュータによる技術計算は1943年の最初の電子計算機であるENIACによる弾道計算からすでに40年の歴史を持っている。近年、ハードウエア技術の急激な進展にともない技術計算の世界も激変しつつあるが、そろそろ技術計算をとりまく全貌がはっきりしてきたし、どのような技術とノウ・ハウが必要かということも分かりかけてきた。したがって、この小冊子では単なる技術計算ということではなくて、題名のとおり、システムとしての技術計算処理全体を考えることとする。ところで、コンピュータによる数値シミュレーションは、たとえていうと、古代・中世の占い師や僧侶の役割を演じる恐れがある。つまり、壮大かつ複雑な計算システムを使って、コンピュータや計算に疎い一般大衆を畏敬させるわけである。

パーソナルコンピュータからスーパーコンピュータや並列計算機までの各種のコンピュータが身近に使えることはよいが、単に道具としてブラックボックスのまま利用するという態度では誤用・悪用を防げない。問題の把握から結果の解釈・評価までのすべてを理解しておくことが非常に大切である。技術計算の処理を図に描くと、図1.1のようになる。つまり、

1．現象の理解
2．現象のモデル化または定式化
3．パラメータまたは数値化
4．データの収集と利用
5．数値計算
6．アルゴリズム
7．プログラム
8．ハードウエア上での実行
9．現象による評価

の9段階を適宜くり返して、必要とする数値シミュレーションが終わる。技術計算システムとしては3－9が関係する。とくに6－8に関係する要因を詳しくすると図1.2となる。これら各段階での問題点をつぎにおおよそ示そう。

図1.1

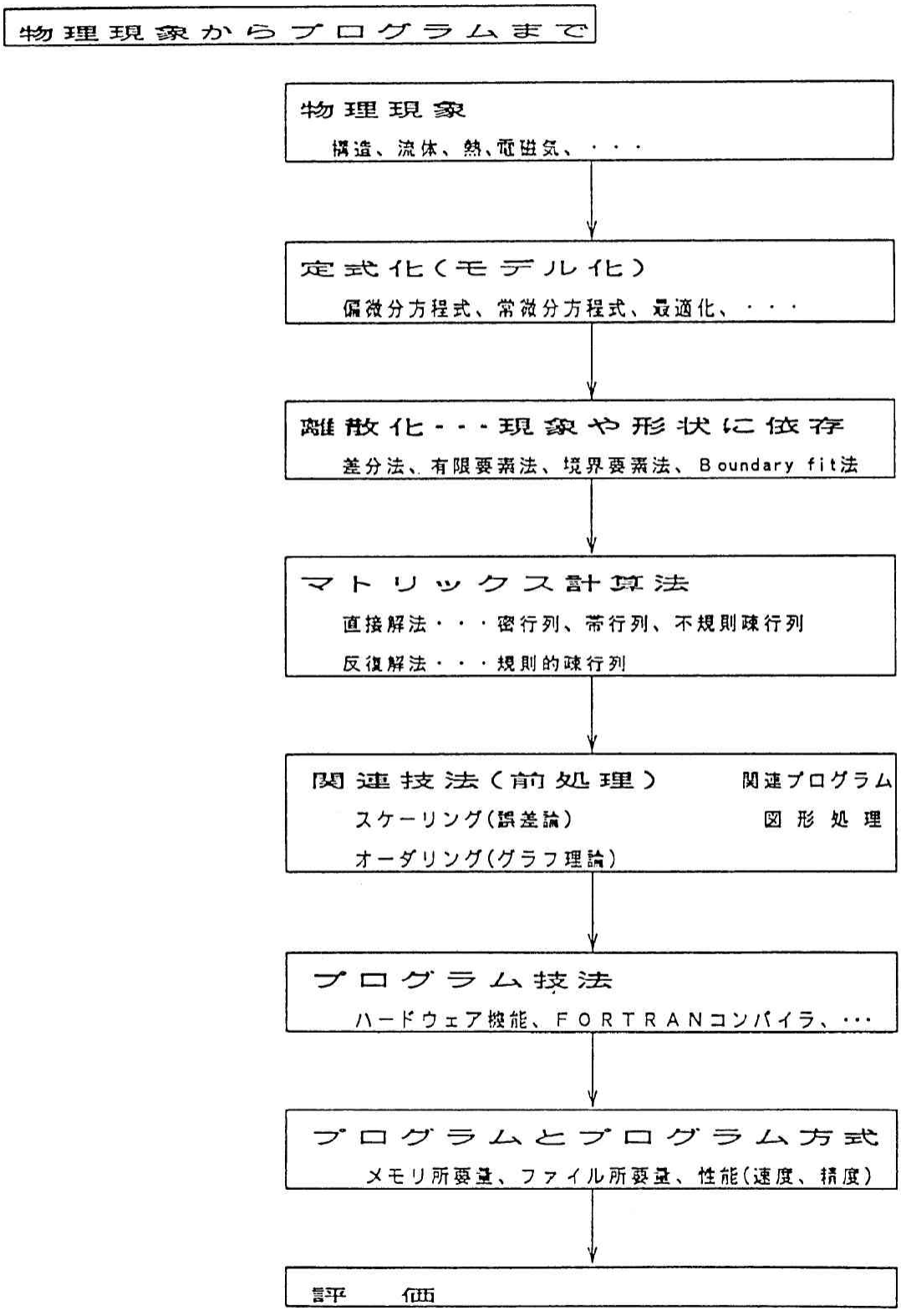
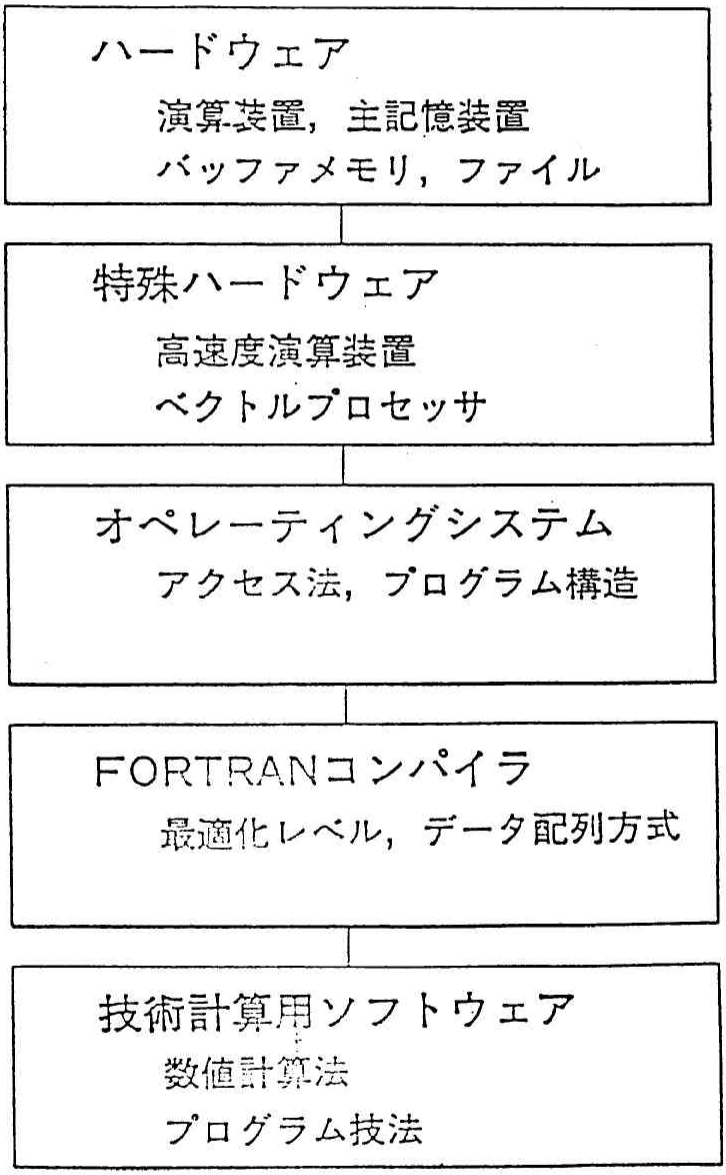

物理現象からプログラムまで
物理現象
構造、流体、熱、電磁気、・・・
定式化（モデル化）
偏微分方程式、常微分方程式、最適化、・・・
離散化・・・現象や形状に依存
差分法、有限要素法、境界要素法、Boundary fit法
マトリックス計算法
直接解法・・・密行列、帯行列、不規則疎行列
反復解法・・・規則的疎行列
関連技法（前処理）
スケーリング(誤差論)
オーダリング(グラフ理論)
関連プログラム
図形処理
プログラム技法
ハードウェア機能、FORTRANコンパイラ、・・・
プログラムとプログラム方式
メモリ所要量、ファイル所要量、性能(速度、精度)
評価

図1.2

性能に関する中間要件



1．1　現象の理解

現象に携わっている人たちは先刻承知のとおりであるが、現象をどう捉えるかにより、つぎの段階からあとの処理が違ってくる。たとえば、きわめて高速に動く物質の運動はニュートン力学でモデル化しても、シミュレーションの結果はあまり精度のよいものにはならないだろうし、非線形現象にやみくもに線形モデルをくり返し適用しても良好な結果が得られるとは思えない。現象を理解するうえで重要なのは現象をよくみて経験を積み、知性を磨いて広い視野に立った判断をすることである。コンピュータが安くなって身近かに普及し、実際の現象と切り離された環境のもとで設計をするようになると、対象が複雑で大規模になるにつれて思いもよらない欠陥や事故が起こることになる。設計者や研究者はともすると独善的になるし、ほかからの批判に対しては防衛的になる。バイオ、医療、原子力に関する安全性などについてもそういう面がいえる。ほかの国や会社の製品とは違って自国や自社の製品だけは大丈夫であるという感情に訴えた話は説得力に乏しい。

原子炉の設計にあたって、事故のとき冷却水による冷却システムがどう働くか、またその安全性はどうなるかを米国でコンピュータ・シミュレーションにより評価した例がある。評価の結果、納得のいく範囲で原子力を建設し稼働させた。数カ月たって心配なので実際に冷却水を原子炉に送って事故を未然に防げるかどうかを実地に試してみた。ところが冷却システムは稼働しなかった。冷却水が原子炉に入ったとたんに蒸気となり、蒸気圧により冷却水が原子炉内には入っていかなくなったためである。このように、現象をどう捉えるかがきわめて大切になる。はじめて設計する巨大システムの場合には不確定要素が山積しており、十分な実験や運転もへずにシステムの安全性を保証するなどということはできない。

ある電力会社を訪問したとき、研究所内に作られたダムの実験用模型を見せてもらったことがある。すべての縮尺を数十分の一にしてあり、湖内の岩まで観測に基づいてそうしてある。しかし、流れる水の分子までは小さくはできない。しかも、形状が同一で大きさが異なるとき、諸変量の間の関係が同じであることは珍しい。普通のトカゲとガラパゴス諸島の大きなトカゲとでは運動量や荷重その他に違いが出てくる。また、洪水にともなう湖岸や川岸の浸透圧などの影響も簡単な模型では知り得ようもない。数値シミュレーションにもこの種の制約はあり、

モデルの制約条件をよく認識して結果を解釈する必要がある。

1．2 現象のモデル化または定式化

物理現象はふつう、偏微分方程式で記述できる。つまり、現象を左右する主なパラメータ間の相互依存関係が定式化できて、攪乱パラメータの影響を無視できるということである。ところが、現象ごとに偏微分方程式の境界条件は違ってくる。現象をどう精密にとり上げるかによって、偏微分方程式の形と境界条件が変わってくるし、時間項や粘性を入れるかどうかによってあとの段階の処理がふつう違ってくる。また、微分方程式を差分方程式として解くときにはカオス現象のような別な性質が現れる。生命方程式の場合には自然界は差分方程式の解に合っていることが近年になって分かった。

ポアソン方程式のように、数多くの物理現象が同一の偏微分方程式に定式化できるときには、この偏微分方程式を理解すると全体の見通しがよくなり、強力な道具になる。輸送、配合、精製などの計画に用いられる線形計画法などもポアソン方程式に似た抽象化の効果があると考えてよい。研究者や設計者が自分の経験した現象と偏微分方程式との関係をもちよってデータベース化しエキスパート・システムとして活用できれば非常に役立つのではないだろうか。できれば、特定の偏微分方程式系に定式化したときの利害得失も明らかにしておけば、あとに続く人たちは多大の恩恵を受けることになる。すべてのモデル化技法は道具であり、道具としての便利さと限界を持っているのである。研究者は研究の新規性を追い求めることに急なあまり、すでに得られているノウ・ハウの蓄積と普及におろそかになりやすい。しかしながら科学の進歩は結局のところ、ノウ・ホワットやノウ・ハウを体系化して事象を統一的に解釈可能にすることにある。

単純な系では主要なパラメータに着目してモデル化しても、現象をよく近似できることが多い。しかし複雑な系では、平常は大した影響がなく無視できるパラメータがシステムを不安定にしたり、大きな変化をとげるきっかけとなることがある。また、システムはその弱い箇所に変化量の増大をまねきやすい。生物システムや社会システムにはこの傾向が顕著であり、最近の東欧の急変はその典型的な例であろう。自然現象の場合でも相変化はその例である。

図1.3

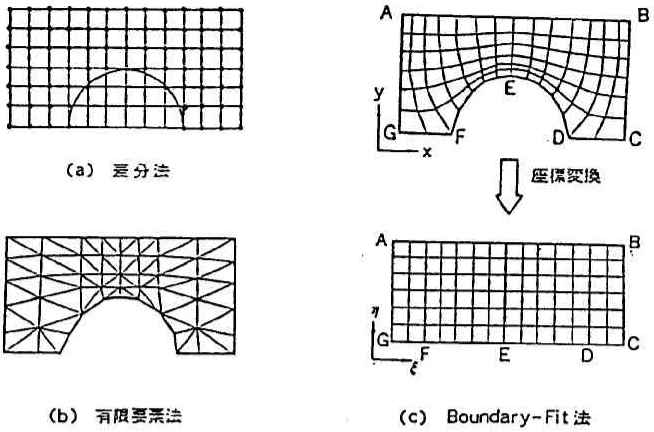
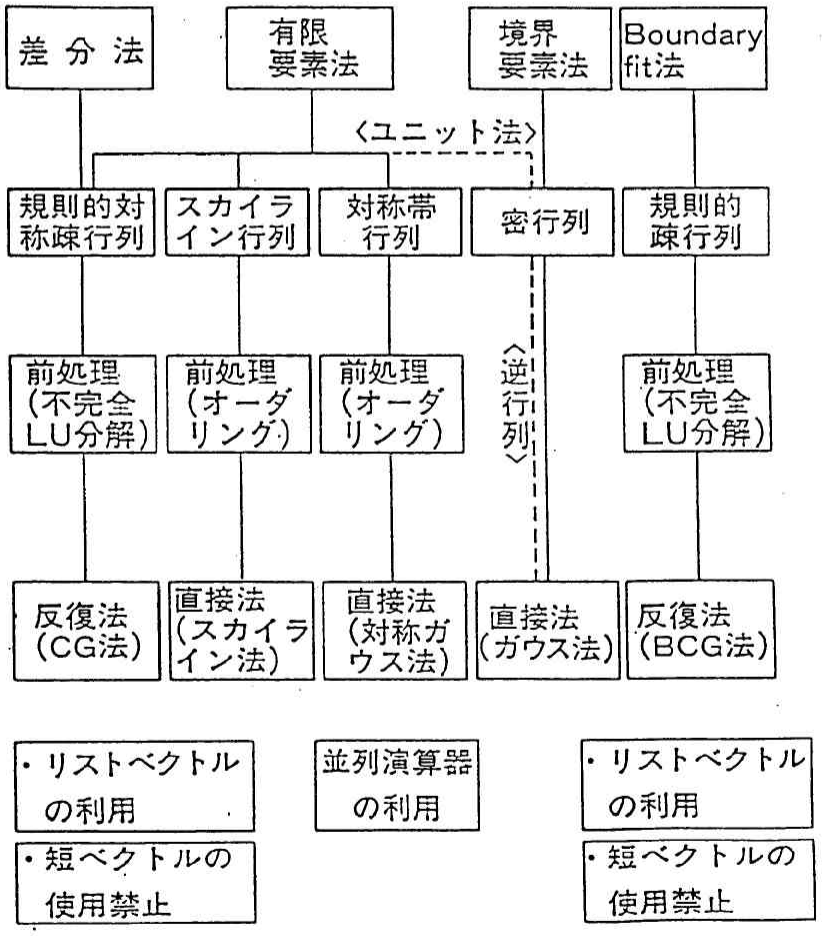


図　座標格子の比較

図1.4

離散化法と行列演算



(注) 部分構造反復法

・・・部分構造ごとに直接法，系全体は反復法

1．3　パラメータ化または数量化

これからの段階が技術計算システムの世界である。あるモデルをコンピュータが処理できるようにパラメータ化または数量化（離散化）するわけであるが、これには数多くのやり方があり、さまざまな要因がからんでくる。

よく知られたとおり、偏微分方程式を離散化するやり方としては、

1．差分法　2．有限要素法　3．境界要素法　4．境界適合法

などがあり、差分法、有限要素法、境界適合法（boundary fitting method）の関係については図1.3に示してある。現象を分析するに当たって、現象の特性、境界の形状、所望の解析精度、計算費用、などによってどの離散化法を採用したらよいかが変わってくる。離散化法が違えば得られる行列の形も違い、図1.4に示すとおり、以後の数値計算法やアルゴリズムも変わってくる。たとえば、ジェット飛行機の翼の周りの衝撃波の解析にはどのやり方でもよいというわけにはいかない。それぞれの離散化法にどういう利害得失があり、どのような現象に向いていて、どのような現象に向いていないかを明らかにしてデータベース化することがとくに望まれる。もっとも、コンピュータ性能（計算速度と記憶容量と転送速度）が現在のスーパーコンピュータの千倍にもなれば、メッシュを細かくして差分法を適用すればよいという意見もある。しかし、20年前にもそういわれたものだ。

離散化には場のメッシュ分割がからんでくる。現在ではグラフィック装置を使った自動分割が前提になっているが、これとても問題がある。現象をよく理解していないかぎり、分割の仕方は機械的なものとなり、計算時間をやたら空費させる結果になりやすい。しかも、自動分割、計算、結果の視覚化が別々の段階として順番にくり返されるときにはこのムダづかいがはなはだしい。一貫した処理の要所要所で途中の過程を把握できれば、誤った処理を最後まで続けることはなくなる。計算効率を上げるために、メッシュをどうやったらよく現象を表現できるかということで格子生成法が工夫されたし、機械的なメッシュ分割による処理時間の増大を防ごうとオーダリング法が使用されるようになったのである。こういった周辺技術に対する認識が薄いことにも問題がある。また、離散化するときに用いる数値積分によっても結果の精度が変化してくる。積分の刻みを細かくすれば精度がよくなるわけでもないのであって、ときには振動したり、発散したりする。したがって、基本的な数値計算上の素養は、道具を使う以上は身につけてお

くべきである。

1．4　データの収集と利用

科学技術計算用ソフトウェアを作成する人間は数値解析とソフトウェアの技術者であることが多い。彼らはふつう現象を扱うことは少ない。したがって、現実のモデルをつくって、それにもとづくテスト・データを準備するということは不可能に近い。せいぜいできることは理論値の分かっている小さなモデルでプログラムの検証をすることぐらいである。つまり、プログラムのテストが十分になされないということである。かりに研究者や実務者がプログラムの作成に協力するにしても、ごく一部のモデルを用意できるにすぎない。このため、良質な科学技術計算用ソフトウェアをつくるには利用者団体の全面的な協力を得ることが必須となる。

作成後のソフトウェアについても信頼性を向上させるにはまんべんなく利用者からモデル・データを集めてデータベースを作るとともに、テストを充実させなくてはならない。物理実験によって実測データが得られているならば、その結果もデータベースに収録し、ソフトウェアの評価に用いるべきである。当然、ハードウェアの場合のようにフィールド・テストとかヒートランといった検証作業が必要である。しかも、これら体系的になされたテストの結果をまとめて文書の形にし、できれば利用者が手引き書として使えることが望ましい。ソフトウェアは多くの機能をもち、その自由に近い組み合わせを許している。また、プログラム上の主記憶に関する制約内ならどんな大きなデータでも処理可能に見える。しかし、これは間違いである。実際にモデル・データを実行した範囲内でしかそのプログラムの機能の正しさを認めることができない。このため、モデル・データの整備と、そのテスト結果の文書化は非常に大切である。

コンピュータのハードウェア性能を測るための標準問題をベンチマークテストとよび、その測定のことをベンチマーキングとよんでいる。米国ではベンチマーク・テストを標準化するためにリバモア・カーネルがよく用いられており、ベンチマーク・テストによってどういう項目を評価したらよいかについても米国商務省標準局からチェックリストが提示されている。よいソフトウェアというものは、プログラム自身はもちろんのこと、データや評価の仕方などに統一がとれている

のである。これらの性能測定用データによる結果も上のデータベースに追加し、活用すべきである。

モデル・データをデータベースに収録するときに、付属情報としてメッシュ分割の仕方、モデルの性格・特徴、特定のコンピュータ（ハードウェアおよびコンパイラのバージョン）での性能結果などもあわせてしまっておくべきである。コンピュータはソフトウェアを含めて製品ではあるが、それを使いこなす技術もそれに劣らない立派な売り物になるのである。これは自動車の場合に運転手という職業があるのと似ている。よい運転手は単なる車の操作者（オペレータ）ではなく、最短の距離と時間および最低の運賃を心がけているはずである。

1．5　数値計算

特定の離散化法でモデルを離散化すると、対応した性質をもった行列が得られる。得られた行列はそれぞれ違った形と性質をもつ。主なものをあげてみると、図1.4に示すとおりになる。つまり、

1．差分法なら、規則的疎行列
2．有限要素法なら、
   a．形状が長方形または直方体に近ければ、規則的疎行列に近い
   b．形状が比較的単純であれば、帯行列
   c．形状が複雑であれば不規則疎行列
3．境界要素法なら、密行列
4．境界適合法なら、規則的疎行列
5．回路解析なら、乱疎行列
6．線形計画法なら、乱疎行列
7．統計解析なら、対称密行列

である。しかし、これは原則にすぎない。流体でも、拡散項だけなら対称行列だし、移流項が入れば非対称行列になる。境界適合法を使うと、差分法でも非対称の疎行列を扱うことになる。しかも、全体を部分構造に分けることができる場合、あるいは分けなければならない場合には、必ずしも原則どおりにはならない。たとえば、境界要素法でもこのときは密行列にはならない。ブロック疎行列の各部分行列が密行列になるというだけである。有限要素法の場合にも、ブロック疎行

列がえられ、各部分行列は密行列になるときもあるし、帯行列や疎行列になるときもある。

数値計算法には計算精度を向上させるための数値解法と、計算量を少なくするためのグラフ理論の応用とがある。スケーリングは前者に関係し、オーダリング法は後者に関係している。反復解法の前処理にスケーリングを使うと、反復計算の回数が減って、全体の計算時間が著しく減ることがしられている。数値解析でスケーリングやオーダリング法を統一的に教えないのは不思議である。これもまた全体を理解している人がいないせいかも知れない。

行列の形は、形状の分割の仕方と節点番号のつけ方に依存し、節点番号はオーダリング法の違いによって変わってくる。あるいは、自動分割プログラムが採用している番号付けのやり方によって左右される。一般に、自動分割プログラムを開発している人たちはこの辺の知識をもっていないから、合理的に番号をつけているとは考えられない。もっとも、現象とは切り離して勝手にオーダリングをするのは、現象の構造を無視して計算効率の観点からだけ自動化するので、問題だとみなす立場もある。計量経済で常識的な変数間の符号の関係を無視して回帰分析結果が得られることがあるが、現象と反した結果は解釈に困ることになる。モデルが不十分だとこういうことはよく起こる。

このように、離散化に当たってはさまざまな事柄が関わりをもち、得られた行列の形により、採用すべき計算手法やアルゴリズムが変わってくる。行列の形に対応する計算法も図1.4に示してあるが、これも原則にすぎない。

1．規則的行列には反復法

2．不規則疎行列には

　a．1行あたりの要素数が少なく、最大要素数も小さいとき、反復法

　b．1行あたりの要素数が非常にばらついているとき、疎行列用ガウス法

　c．上の二つの中間に対しては、ガウス法

3．密行列や帯行列に対しては、ガウス法

たとえば、回路解析では、一般に疎行列ガウス法であるが、過渡応答解析のように数百回もデータを一部修正して解きたい場合にはガウス法と反復法を適当に組み合わせて使うことも考えられる。また、接触問題では逆行列の一部の修正をくり返すことが必要となり、一度だけ連立一次方程式を解く場合と計算法の選び方

表1.1

スケーリング(SCALING)とオーダリング(ORDERING)

スケーリング

目的：数値計算を安定にする

効果：計算精度がよくなる

反復法のとき速度が上がる
（アルゴリズムが単純）

オーダリング

目的：計算量を少なくする

効果：直接法のとき速度が上がる

直接法のとき記憶容量が少なくてすむ

反復法のとき配列の次元下げが可能

部分構造化の自動化が可能

欠点：直接法のとき計算精度が悪くなるときがある

大規模行列のときノウハウが必要

→モデル化と関連する

行列の形と計算法

１．規則的疎行列には反復法

２．不規則的疎行列には
- a. １行あたりの要素数が少なく、最大要素数も小さいとき、反復法
- b. １行あたりの要素数が非常にばらついているとき、疎行列用ガウス法
- c. 上の二つの中間に対しては、スカイライン法によるガウス法

３．密行列や帯行列に対しては、ガウス法

ただし、

１．スケーリング法やオーダリング法

２．A(x)＝b(t), t＝1, 2, … を少ない時間で解きたいときは、ガウス法

３．A(t)x＝b(t), t＝1. 2, …,
A(t)の構造とb(t)の構造はtに独立のときには、コード生成が有利（除くスーパーコンピュータ）。

が違ってくる。さらに、１．の反復法にも行列の形に対応して各種の方法があり、収束の仕方は特殊な場合以外はやってみないと分からないという状況にある。

反復法は行列の性質によって計算時間が非常にかかるし、１行あたりの非ゼロ要素数がばらついているときに用いるとガウス法より時間がとてもかかる。プログラムにしろ、数値計算の本にしろ、こういった実用上の制限についてはあまりふれていない。プログラムを組んだり、数値解析の本を書いたりする人は、必ずしも最低限度必要な範囲の問題を試していないからである。むしろ試すこと自体が無理である場合が多い。さまざまな現象に実際に接している研究者や設計者ならテストすべき問題を系統的にとり上げることができようが、筆者のように現象に携わっていない者にとってはすぐに限界にぶつかる。しかし、道具は得手不得手をはっきりさせなければ危なくて使えない。こういうわけで、テスト・データのデータベースが切実になってくる。

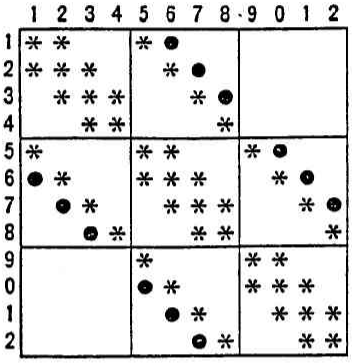
オーダリング法にしても、単一構造の場合はＲＣＭ（reverse Cuthill-Mckee）法などでうまくいくが、部分構造からなる複雑な構造物にそのままＲＣＭ法を適用してもうまくいかない。まず、全体をブロック化して部分構造に分けたあと、それぞれの部分構造ごとにＲＣＭ法を適用する、といったことが望ましい。アルゴリズムにしろ、数値計算にしろ、ある制限条件のもとで、制約された範囲内で活用できるのである。はやり風邪のようにブームになっている手法を無条件に採用するなどといった習慣は身につけるべきではない。数値計算にも道具としての安全対策が必要なのである。　反復法は主記憶内におさまる範囲の問題にうまく使えるが、補樹記憶を使うような大きな問題には向いていない、こともあまり知られていない。

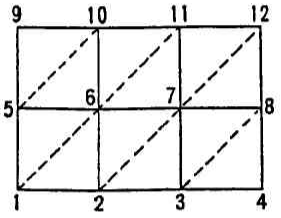
１．６　アルゴリズム

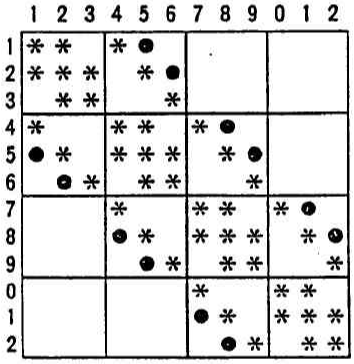
あるパターンをもったデータに対しある数値計算法を特定のコンピュータ上で性能よく稼働させるための具体的な計算手順をアルゴリズムとよんでいる。したがって、数値計算法とアルゴリズムとハードウェアは切りはなせないつながりをもっている。図1.6に示すように、回路解析でのコンピュータ利用はその典型である。ＩＢＭ社が仮想記憶方式のコンピュータを1960年代半ばに生産するまでは密行列ガウス法やそのメモリ節約版の疎行列ガウス法が用いられていた。70年代に

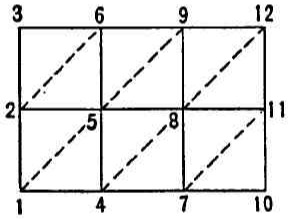
対称行列のオーダリング

図1.5

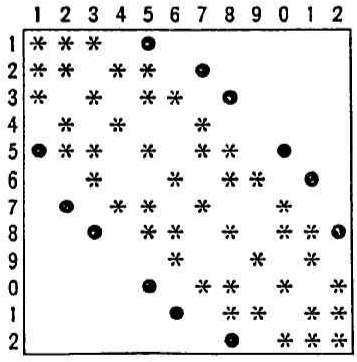


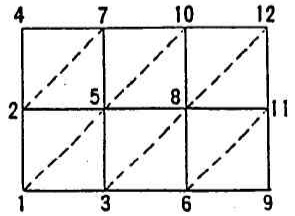






(Cuthill-Mckee法)





なると、仮想記憶に適したコード生成方式のアルゴリズムが採用され、同時に計算量を小さくするためのオーダリング法が利用された。80年代にスーパーコンピュータが利用されはじめると疎行列ガウス法が復活し、過渡応答解析では反復法との併用もなされている。似たようなことは構造解析の分野でもいえる。いずれにしても、コンピュータを使うかぎりはハードウェアの性格に応じたアルゴリズムを採用すべきで、並列計算機になればなおさらのことである。また、アルゴリズムは基本ソフトウェアにも依存し、同一の計算法でもコンピュータやFortranコンパイラの性格を考慮するかどうかで性能が二倍程度違ってくることはよくある。なるべく特定のコンピュータだけに向いたアルゴリズムを避け、広くコンピュータ世界で役立つアルゴリズムを採用することが望ましい。この点で、ベクトル計算機用のアルゴリズムはプロのアルゴリズム屋に任せるほうが無難である。並列計算機用のアルゴリズムについても同様のことがいえる。ただ、これは相対的な話であって、将来やってくると思われる並列計算機の時代には、ある程度の並列アルゴリズムは研究者すべてが理解するようになるかもしれない。とはいえ、現在のベクトル計算機はあまりにも小手先の芸を必要としているので、あまり感心できない。

アルゴリズムの中で、意外に軽くみられているのがデータの整列や検索である。データを記憶すると、そこには何らかのデータ構造が発生する。特定のデータ構造には特定の整列用アルゴリズムや検索アルゴリズムが向いている。半導体の設計のためのエレクトロニクスＣＡＤの世界のように技術計算でも大規模なデータベースを扱うようになってきたが、この辺の技術はさして重視されていないようである。アルゴリズムは主記憶だけの場合、拡張記憶を使う場合、ディスクを使う場合、ベクトル計算機や並列計算機の場合、などはソフトウェアの特性によって変わるのが普通である。

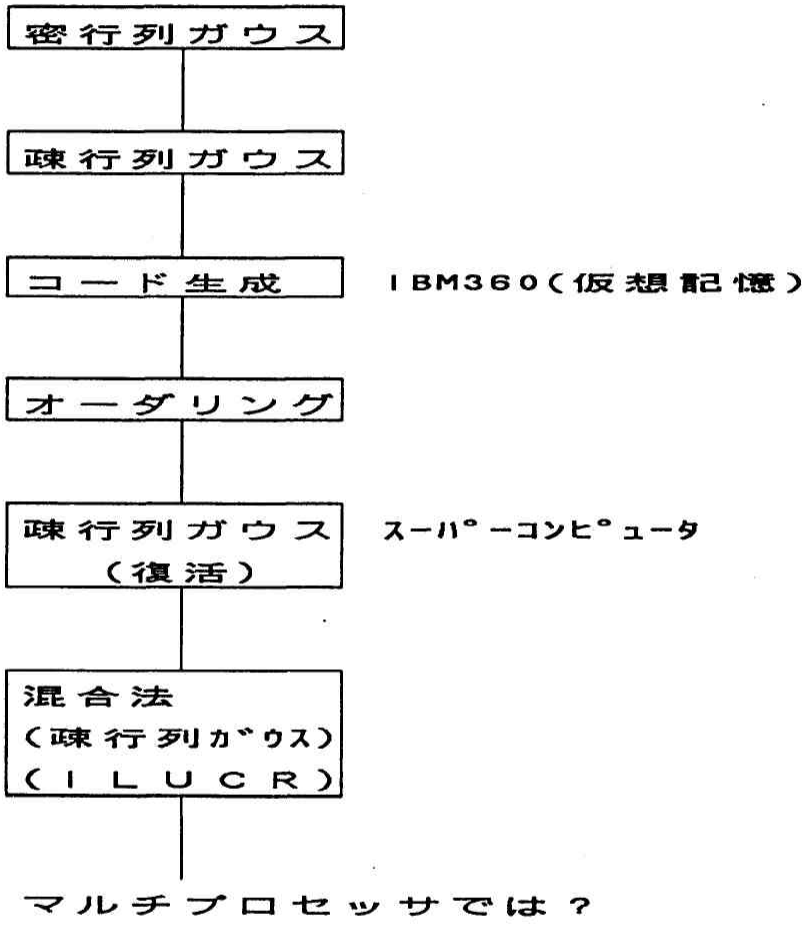
１．７　プログラム

プログラムは実行すべきコンピュータやコンパイラの特性を考慮して作成するときに性能がよい。しかし、他のコンピュータで実行するときにも性能が劣化しないという広い意味での移植性が望ましい。もっとも、ハードウェアに依存して核の部分をサブルーチンの形でハードウェア・メーカがコンパイラの一部として

図1.6

アルゴリズムの発展

回路解析



提供してくれればこういったことはかなり防げる。つまり、利用者はこの核部分を必要とするたびにプログラムの中から呼びだせばよい。しかし、ハードウェア・メーカはこのような基本となる部分を提供していないのが普通である。ハードウェア・メーカは基本的にハードウェアの売上を増やすことだけに興味があるにすぎない。たとえば、配列のゼロクリアにしても、本来ならそれにふさわしいハードウェア命令を使うべきであるのに、Fortranコンパイラは配列の各要素を一つずつゼロにおきかえているというようなことが起こる。利用者がコンピュータをよく知っていて、不具合な箇所を指摘しないかぎり、ハードウェア・メーカ側は意識的にか、無意識的にか、とにかく大して改善してくれない。官公庁に対するコンピュータ・システムにおけるソフト的な部分の一円の入札がその端的な表れで、本質的にはソフトウェア的なものを軽視しているのである。

プログラムは、建造物と同じく、部品、モジュール、ユニット、というように階層的に組み立てていくのがよい。このとき、部品やモジュールやユニットといった構成要素が標準化され、性能面と信頼性で心配のないように、しかも系統的に広い用途にあわせて作成されていなければならない。ところが、こういった部品やモジュールは新しいコンピュータが売り出されるときに、一時的にハードウェア・メーカによって開発されるだけで、あとは凍結状態になるのがごく当たり前になっている。国内のコンピュータ技術がハードウェアと、せいぜいオペレーティング・システムのような重く堅いソフトウェアどまりになっていて、技術計算に必要な数値計算用部品がどこからも供給されないため、米国や英国の製品に席巻されてしまっている。それだけソフトウェア産業が弱いといえる。部品の品質や性能をブラックボックスのままにして使って事故が起きたとしても、その責任から利用者は逃れることはできないはずであり、無関心でいてよいわけではない。利用者が何らかのアクションがとれる状態を作っておかなくては数値シミュレーションの発展は心細いものとなる。

プログラムにバグ（不良）はつきものである。このようなバグをもったプログラムのバグの箇所を見つける（作成者側）ことと、計算結果の良否を判断する（利用者側）ことが常に必要である。また、そのための治工具やデータの整備が必要となる。とくに、新しい修正版のプログラムを使うときには受け入れテストがぜひ必要である。普通バグとはみなされないものに、ひどい性能と計算精度と

がある。しかし、性能と計算精度についても品質を保証するにはどうしたらよいかといった技術も大切である。

プログラムはふつう特定の離散化法と数値解法だけを用意して作られており、モデルに応じていくつかの離散化法と数値解法（アルゴリズム）を選べるようにはなっていない。図3.6に示すように、問題の規模が変化してもアルゴリズムの優劣は違ってくる。したがって、ある特定のアルゴリズムだけしか持っていないプログラムは利用者に時間的費用的な負担をかけていることになるのである。年々、ハードウェアの性能が向上して値段が安くなり、人件費が相対的に高くなる一方だし、数値計算法の評価がまじめになされていないので、メーカも利用者もこのあたりの感覚を欠いているのかも知れない。

1．8　ハードウェア上の実行

プログラムを特定のハードウェア上で実行するわけだが、これにはデータの作成および入力から実行をへて結果の出力および編集までの処理がある。

構造や流体では対象物をワークステーション上の図形処理にしたがった自動分割により離散化し、できたデータをファイル経由でホストコンピュータ上の解析用ソフトウェア（ソルバという）に入力する。遠隔地からのデータ転送の場合には通信回線の速度がネックとなって、応答が悪く、円滑な処理ができない。自動分割もホストコンピュータでやるときには負荷が高く悲劇的になる。

現在ではホストコンピュータが１台しかない場合がほとんどであるからさして問題はないが、将来はコンピュータ・ネットワーク化されて、図4.1に示すようにネットワークには各種の汎用コンピュータ、ベクトル計算機、並列計算機（パラレル・コンピュータ）、スーパーミニコンなどが接続されるようになる。しかも記憶装置についても拡張記憶や階層別記憶装置が利用できる。さらに、自宅や職場ではパソコンやワークステーションによりかなりの処理を手軽に進めることができる。このようなとき、どのコンピュータで処理すれば妥当な時間でしかもなるべく安く使えるかが関心を集めることになる。各コンピュータの性能データがデータベース化され、ＡＩ機能が利用できれば、応用ソフトウェア自身が最良なコンピュータを選択できる。少なくも利用者が自分で選ぶことができる。目的地にいくのに、さまざまな交通機関と経路があるとき、利用者が自分にあったもの

を選ぶのと同じである。このとき、利用者がプログラム媒体やプログラム自身をいちいち変換しないで済むことが前提である。現在のように、互換性に乏しい端末装置やホスト・コンピュータではコンピュータ・ネットワーク化もあまりうまくいかないだろう。

こういった選択が理論上許されるようになったとしても、実際上はコンピュータの性能や処理費用の算定に必要な基準データがない。ハードウェア・メーカもソフトウェア・メーカも客観的なデータを集め、公表することにきわめて熱心でない。米国ではハードウェア、ソフトウェア、アルゴリズムの一連の関係を標準テスト問題で測定することが重要視されており、この作業をベンチマーク・テストとよんでいることはすでに述べた。特定のコンピュータ向けのコンパイラは、ほかのコンピュータと言語仕様が同じであっても、その最適化の仕方はハードウェアやコンパイラ設計者の技術力とに依存するので、性能はかわってくる。したがって、ベンチマーク・テストを実行しても、必ずしもハードウェアの優劣だけを評価したことに成らず、ハードウェア、オペレーティング・システム、およびコンパイラを総合的に評価したことになる。このため、メーカの総合力が表に出てしまうので、評価に対しては消極的になる。

1．9　評価

特定の応用ソフトウェアを用いた数値シミュレーションでは、現象の理解、モデル化、離散化、数値解法、ハードウェアの各段階で誤差を生じ、不確実性を産むことになる。このうち、モデルと数値解法上の不安定要因については図1.7に示してある。これらの誤差の評価についてはいろいろな感度分析が必要となる。したがって、評価の効率は、利用者の現象に対する経験が豊かであるかどうかにより良否が決まる。現象をよく知らない利用者は現象に関係する各種の要因（パラメータ）を動かして、シミュレーションの安定性をはかることになる。その際、ゆき当たりばったりに要因を変動させてもダメで、図1.8に示すとおり実験全体をうまく制御しなければならない。そのためには実験計画法などの統計的手法や確率論に対する素養が必要である。また、実験の結果から有意な判定を下せるかどうかを知るためにも統計的な手法が必要になってくる。普通、理論値と実験値とを図に表示してすませているが、あれではあまり客観性がない。

図1.7

数値シミュレーションの不安定性

・モデルの不確実さ
（1）当該物理理論とデータが正しくないか
（2）数学モデルが十分正確に現実に反映していない
（3）物理問題が本質的に悪条件であるか
（4）不安定な計算がなされるような数学モデルを作った

・数値的不確実さ
（1）丸め誤差の影響
（2）アルゴリズムが数値的に不安定
（3）手法が早めに収斂した

・ソフトウェアの不確実さ
（1）プログラムのへま
（2）数値解法・アルゴリズのライブラリがよくない
（3）コンパイラの誤り
（4）データの不備
（5）悪いプログラム

・感度分析と誤差評価
（1）数値データ　：データに摂動をかける
（2）丸め誤差　：ビット長を変えて同じ結果を得る
（3）使用した方法：手法を変えてランする
（4）モデル　：モデルの修正

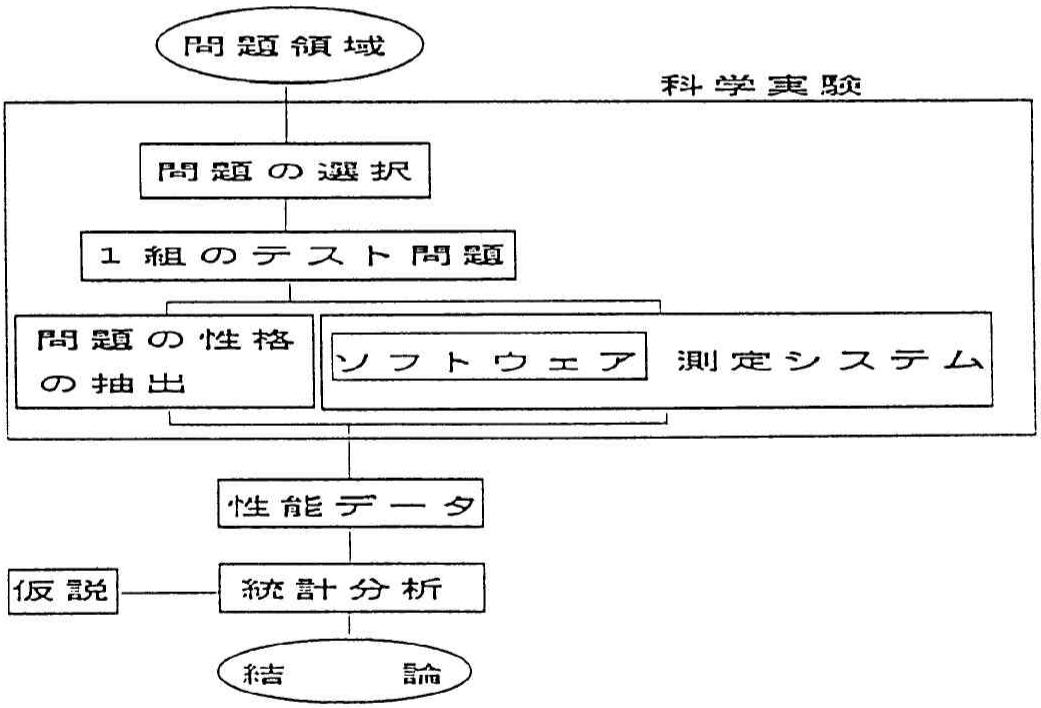
図1.8

ソフトウェアの性能評価(Rice教授)

・方法

1．統制がとれ、うまく設計された実験の枠組
2．ちゃんと定義した性能測定の使用
3．はっきり記述された評価基準
4．性能データについての完全な情報の提供
5．体系的ないし統計的手法を用いてデータを分析

・手順



○ アルゴリズムの差よりコンパイラの差の方が性能に影響を及ぼす

使用する応用ソフトウェア自体を評価することも大変大切である。とくにすぐれたアルゴリズムをもっているか、移植性はどうか、といった点に注目すべきである。また、実験結果の文書化によりノウ・ハウの蓄積をすべきであり、問題点や失敗例を他の利用者のためにも残すべきである。ソフトウェアの評価のためのチェックリストを図1.9に示してある。

数値シミュレーションの結果は物理的な実験によって検証すべきである。モデル化の不完全さは数値シミュレーションそのものによっては見つけることができない。飛行機や車のまわりの流れの可視化がどんなにすばらしくできても、実際に風胴実験をしたり車を走らせてデータを収集し、計算結果とくらべてみなければ何ともいえないのである。数値シミュレーションによって物理実験にともなう時間や費用を減らすことはできても、ゼロにできるわけではない。評価をきちんとするには、図1.5に示したように、物理実験から得られたデータを系統的にデータベースにしまっておくことが必要である。普通データベースというと、形状データや材質データを意味するが、実験データのデータベースもまた非常に重要である。回路の論理設計で用いられるルールも経験上得られた特性データのノウ・ハウによるのである。

以上にふれた８つの段階の問題点を解決するにはどうしたらよいか。これを考えたのが本冊子の技術計算システムである。その一部は筆者の個人的な見解にすぎないかもしれないが、おおよそは5－10年後に実現していくものと思われる。そういう意味ではシミュレーションはこれから新しい活気に満ちた時代に入っていくといえよう。

図1.9

ソフトウェア評価のためのチェック・リスト

・アルゴリズム：扱う問題とアルゴリズム
1．複雑さ　2．収束性
3．誤差評価　4．ステップ当りの処理量

・プログラムと計算環境
1．言語とコンパイラ　2．ＯＳ
3．ハードウェア　4．入力形式
5．プログラムの可用性

・実験の設計と結果
1．実験対象　2．手順　3．可能範囲
4．前処理　5．後処理
6．結果の記述
7．母集団と標本法

・結果の報告
1．実験の設計と手順の完全な記述
2．性能測定　3．正当性
4．プログラム・パラメータ
5．終了条件　6．許容誤差　7．失敗例
8．測定法などの実験要因を変えたときの効果

・結果の記述
1．予想と結果の違い
2．仮設の記述
3．将来の研究とアルゴリズムの改良
4．興味あることが観測された場合その特殊な問題の指定

図1.10

1）数値シミュレーション用ソフトウェア

2）数値シミュレーション技術

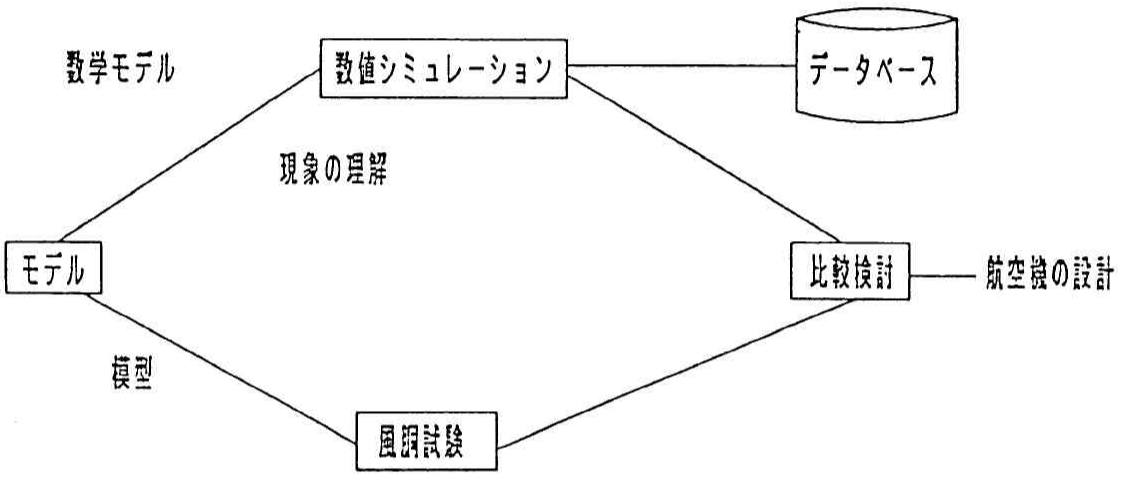


図　　数値シミュレーション風洞の位置付け

2．これまでの科学技術計算の歩み

先を読むには、歴史を知る必要がある。先人たちが乏しい資源を効率よく、うまく使おうとした努力は時間とともに埋もれがちであるが、知恵は乏しい環境の中で生まれやすいものであり、先人の努力を知ることがぜひ必要である。

技術計算とコンピュータの関係のごく初期は表2.1に示した経過をたどる。また、その後の歩みを箇条書きにしたものが表2.2である。ヘルマンによる積分器は今からおよそ180年前に作られている。偏微分方程式を解くために用いられている差分法もまた160年近い歴史をもっている。モデル化の最初の失敗はドイツのシアッチが長距離砲を今世紀はじめに設計したときに起こった。シアッチが計算した距離より大砲の飛んだ距離は２倍にも達したが、これは大砲の弾が空気の薄い抵抗が半分の高さを飛んだためである。ついで、1910年５月に気象学者のリチャードソンは偏微分方程式にもとづく上空４層の数値予報モデルを作り、大陸を一辺が200kmの格子状に分割し、各節点に妙令の女性を配置して計算を同時にやらせ、結果を電話で連絡させて次のステップの計算を指示した。結局はモデルの不備と計算間違いにより失敗してしまったという。つまり、リチャードソンの夢として名高い話である。これに対し、1946年に開発されたENIACでは一辺の長さを736kmとし、時間の刻みを３時間にして24時間の予報を同じ24時間で計算した。現在ではスーパーコンピュータにより流体計算が可能になり、それなりの衝撃を与えているが、手計算に替わるコンピュータの出現はいまよりはるかに強い衝撃と、その利用の仕方についての研究心をよび起こしたといえよう。現に、マトリックス計算に付随した本質的な話題は1950年代と60年代とでほぼで出つくしており、今日にいたる進歩はその改良がほとんどである。

1951年にアメリカ商務省標準局主催で第一回の数値計算シンポジウムが開催されたが、日本の場合と違って、標準局はコンピュータ利用に関して今にいたるまで主導的な役割をはたしている。標準局は、さまざまな標準化を推進するだけでなく、人工知能としてのロボットについても特色のある研究を進めている。行列計算は線形計画法とともに発展し、ついで回路解析とともに発展した。1957年にはＩＢＭ社のバッカス博士により科学技術計算用言語のFortran言語が開発され、いまにいたるまで使われている。そして、1970年代にはいると数値計算量の増大

表2.1

数値計算の経過

１８１４年	ヘルマンによる積分器
１８３４年	航海暦および天体暦公刊に際し、差分法
第一次対戦前	ドイツがシアッチの方法によって長距離砲を設計（弾道計算）
第一次対戦中	リチャードソンの夢
１９２０年代	モールトンは差分法を弾道計算に導入（連立常微分方程式）
１９３０年代	ガウス法による連立一次方程式の解（アタナソフ）
１９４０年代	エッカートによる天文学計算における電気式会計機の利用
１９４４年	エイケン／ＩＢＭは自動逐次制御計算機によりベッセル関数の計算
１９４６年	ＥＮＩＡＣで数値予報、弾道計算など
１９４６年	流れ図（ゴールドスタイン）
１９４７年	プログラミングとコーディングの概念（フォン・ノイマンとゴールドスタイン） コーディング・・・数学の言語をマシン・コードへ翻訳
１９４７年	行列の条件数（ゴールドスタイン）
１９５１年	ＵＣＬＡ内の合衆国標準局で「連立一次方程式および固有値計算に関するシンポジウム」・・・ヤコービ法の再発見
１９５２年	最初のコンパイラ（スイスのルティスハウザ） HestenesとStiefelsによって共役勾配法が開発された
１９５４年	ギブンス法
１９５４年	FORTRAN発表（バッカス）、１９５７年に完成 47個のFORTRAN命令文からなるプログラムを４時間で作成し、機械語約1000命令に約６分でコンパイル
１９５８年	ハウスホルダ法
１９６０年代	疎行列用の積形式によるガウス・ジョルダン法が線形計画法で用いられた。また、オーダリングやスケーリングも使用されはじめた。 ＩＢＭがサイエンティフィックサブルーチンパッケージＳＳＰを発表
１９７０年代	ＮＡＴＳ計画が開始され、数学ソフトウェアの質が定義された。 ＩＭＳＬやＮＡＧが市販され、ＬＵ分解によるガウス法が主流となった

表2.2

応用ソフトウェアの経過

1．軍／大学／RAND社（1940年代－1950年代）
(1)弾道計算
(2)航空機設計・・・流体計算
(3)気象・・・数値予報（プリンストン高級研究所）
(4)原子力計算
(5)作戦計画・・・輸送・調達・計画
(a)数理計画法
(b)品質管理
(6)統計学（フォン・ノイマン）・・・モンテカルロ法
(7)数値計算
(8)天文学（エッカート）
(9)フライトシミュレーション（フォレスター）
(10)オンライン・システム・・・防空システムSAGE

2．ＮＡＳＡ／大学・研究所（1960年代－1970年代）
(1)構造解析・・・有限要素法
(2)数値制御
(3)図形処理
(4)知識工学
(5)計量経済モデル
(6)社会・環境モデル
(7)ＣＡＩ
(8)回路解析
(9)統計予測

3．民間／大学（1980年以降）
(1)境界要素法
(2)画像処理
(3)経営計画（意思決定支援）
(4)エキスパート・システム
(5)分子軌道計算
(6)流体計算
(7)資金運用計算（ポートフォリオ分析）

と品質向上を解決するために米国科学財団のNATS（National Activity to Test Software）計画が始まり、その成果として品質の高い数値計算用部品集の作成がなされた。

国内ではこの50年代と60年代に研究があまりなされなかったがために、欧米の「ＳＩＡＭ」のような専門雑誌も発行されることなく、また、専門の技術者が市民権を得ていないわけである。ようやく今年になって専門の学会が設立されて「ＳＩＡＭ」の日本版が発行されることになった。

ところで、技術はつぎのような経過をへて一生を終える。

（１）夢の時代
（２）不遇の時代
（３）華やかな時代
（４）あって当たり前の時代
（５）忘れ去られる時代
（６）他に利用される時代

つまり、50年代のコンピュータはRAND社とプリンストン高等科学研究所を中心とし、60年代のコンピュータはＭＩＴを中心とした夢の時代なのである。科学技術計算用ソフトウェアとして大きな影響を及ぼしたのは構造解析と数値制御であった。構造解析はＭＩＴのICES（Integrated Civil Engineering System）とNASAのNASTRAN（NASA Structural Analysis Sysytem）であり、数値制御はイリノイ大学のＡＰＴ（A Programming Tool）である。この頃は、1957年のソ連による人工衛星スプートニクの打ち上げ成功が米国内に衝撃を与え、ロケット打ち上げや社会基盤のための高速道路の整備が社会の最優先になった時代である。あるいは高速道路整備が軍事的な要請に基づいてなされたのかもしれない。大戦後半のルーズベルト大統領によるテネシー渓谷のＴＶＡ計画でさえ、現在では、原爆用の原子炉建設のための電力を提供するためだったといわれているくらいである。

科学技術計算用ソフトウェアの作成方法に最も影響を与えたのはICESである。1960年代後半に設計されたICESでは、工学の諸分野の人たちが一つのシステムを使えるように次の方針をたてた。

１．異なった性格とデータをもつ数多くの異種の工学的問題を十分に解けるだ

表2.3

ＭＩＴ（マサチューセッツ工科大）のＩＣＥＳ
（Integrated Civil Engineering System）

・高速道路のインターチェンジの設計は multi-disciplineな技術

（1）高速道路工学…高速道路の位置と設計
（2）土壌力学…定着，安定および基礎条件
（3）構造工学…高速道路用橋梁
（4）水力工学…排水
（5）輸送工学…トラフィックの流れ

・ＩＣＥＳに対する要求

（1）異なった性格とデータをもつ数多くの異種の工学的問題を十分に解けるだけの柔軟性をもつこと
（2）エンジニアが自分の問題を適切に構成して最良の解法を選べること
（3）要求された操作を実行できること
（4）工学上の意思決定過程を乱さずに、この過程を推し進めてエンジニアが他の解を検討したり、発見できるようにすること
（5）すべてのコンピュータによる制約を除去して、すべての工学上の制約を表現できること

・ＩＣＥＳが実現した技術

(1)工学の各分野の計算に対応して、各サブシステムを用意した

構造解析　…STRUDL
幾何計算　…COGO
プロジェクト管理…PROJECT
データベース　…TABLE
道路計算　…ROADS

(2)システム記述言語としてICETRANを開発しプリコンパイラ技術を確立した

(3)各サブシステムに共通なデータベース

(4)エンジニアが理解しやすい問題向き言語(Problem Oriented Language)とタイムシェアリングシステム

(5)図形出力

(6)テレプロセッシング(Teleprocessing)のような遠隔計算機能を通じて大型計算機を使うことができる

(7)サブシステムを統合する制御部分(Exective)の機能

(a)コマンド定義……サブシステムごとに違う問題向き言語

(b)コマンド解釈

(c)共通モジュール…領域管理，入出力管理

けの柔軟性をもつこと。

2．エンジニアが自分の問題を適切に構成して最良の解法を選べること。

3．工学上の意思決定過程を乱さずに、この過程を推し進めてエンジニアが他の解を検討したり、発見できるようにすること。

4．すべてのコンピュータによる制約を除去して、すべての工学上の制約を表現できること。

このためのソフトウェア技術として、システム記述言語、プリコンパイラ方式、リスト構造による工学的データベース、問題向き言語、テレプロセッシング、動的記憶管理、図形処理、といったものが採用されている。

最近では数値計算にも関係してきたＡＩもこの頃にＭＩＴのマッカーシ博士が唱えているし、その道具として記号処理言語LISPが開発された。数値シミュレーションもフォレスター教授がDYNAMOを道具として指導し、のちにローマクラブが世界モデルのシミュレーションに採用し、その成果として「成長の限界」を出版した。並列処理に欠かせない多重タスクという考えもＭＩＴの基本ソフトウェアMultics (Multipled Information and Computing) システムを研究している人たちが打ち出しているし、タイムシェアリング・システムは上述のマッカーシが考案した。このように、1960年代のＭＩＴは、今日使っているコンピュータ技術の考え方に大きな方向を与えているのである。次世代の技術計算システムは、60年代のＭＩＴにつづく第２世代の技術計算システムといえるかもしれない。

ソ連のスプートニクに追いつくために設立されたNASA（米国航空宇宙局）では人工衛星を打ち上げるために、さまざまな技術が必要となり、そのひとつに有限要素法を用いたFortranプログラムNASTRANが1971年に開発された。NASTRANは当時アメリカの大型コンピュータであったIBM,UNIVAC、およびCDCのいずれの計算機でも稼働することを目的とし、移植性を第一に考え、長期にわたる使用を第二に考えた。事実、それから30年、主要な計算機でいまだに広く世界中で使われていて、自動車会社の90％は顧客であるといわれている。さらに利用者の便宜をはかるために、使用説明書だけでなく、例題マニュアル、理論マニュアル、プログラマ用マニュアル、応用マニュアルを用意した。プログラマ用マニュアルにはNASTRANシステムに利用者固有の機能を追加するための方法が記述されており、応用マニュアルにはハードウェアごとの使用法や性能について記述してある。NASTRANにより、

表2.4

ＮＡＳＡのＮＡＳＡＴＲＡＮ
（NASA Structural Analysis System）

ロケット開発のための構造計算を速く、精度よく実行するためのプログラム。

・文書の整備
（1）理論マニュアル
…数値解法とアルゴリズム
（2）利用者マニュアル
（3）応用マニュアル
（4）例題マニュアル
（5）プログラマ　マニュアル

・財産としての応用ソフトウェアの開発
（1）1966年に開発開始
（2）1968年静的解析部分の予備版稼動
（3）1969年NASAセンタに引渡し
…field test
（4）1970年末にレベル1.1を一般公開
（5）1974年レベル15.5.1完成…100K steps
（6）1975年レベル16完成
（7）現在レベル19　…500K step ups

・ＮＡＳＡのプロジェクトの進め方
（1）委員会による利用者要求の把握と評価
（2）予備設計，つまり、概念設計
（3）開発。この段階でも利用者からの評価をを吸収し、利用者教育を重視する
（4）利用者団体への引渡し
（5）保守および満足しうる範囲までの改良

ソフトウエアには文書がいかに重要な意味をもつかが明らかとなった。同時にNASTRANは７カ所のNASAセンタにおいてプログラムの実地検査が行われ、テストの重要性も明かとなった。

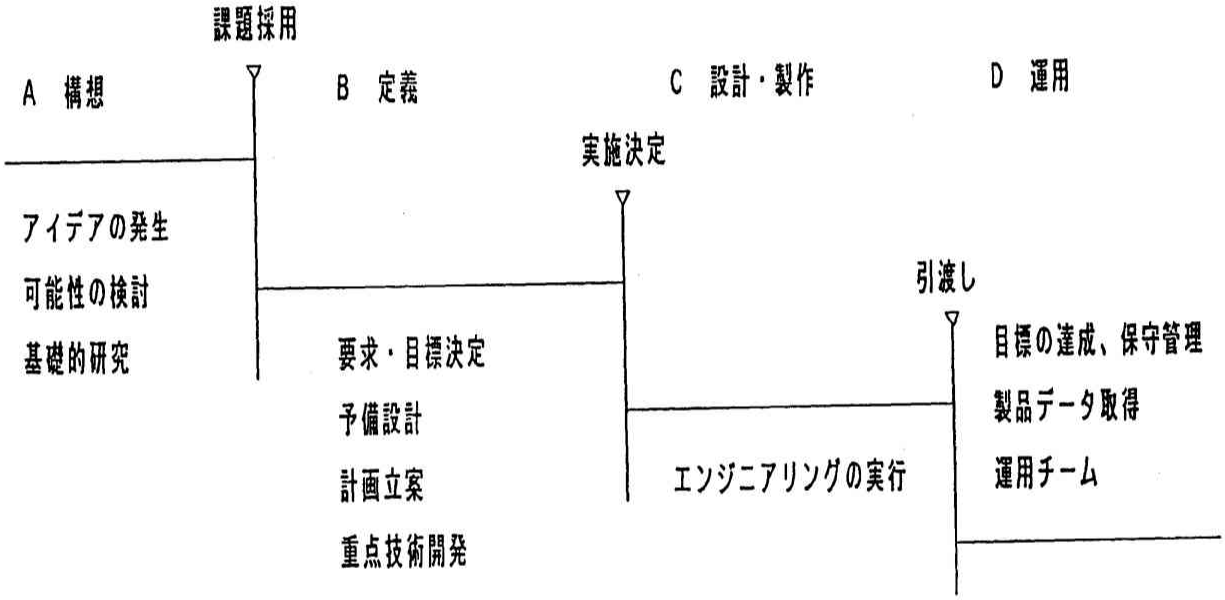
科学技術計算も目的とする物理現象や工学領域が広がり、手段とするコンピュータ・システムが複雑となるにしたがい、技術・管理両面について統率できる統括者の養成と、長期にわたって利用する科学技術計算システムの構築が課題となる。NASAでは統括者の養成のためのシミュレーション・プログラムとしてGREMEXを開発し使用した。統括者は単なる管理者とは違いプロジェクトの進行において適宜意思決定をしていかなければならない。また、NASAにおけるプロジェクトの進め方は図2.1に示すとおりである。構想の段階での充分な検討、定義の段階での評価委員会の役割、設計・製作段階でのプロジェクト自身のフェイル・セーフ、運用段階での10年以上にわたる改良などが特徴である。最近コンピュータ業界で注目されてきたシステム統括(System Integration)の先例がすでに60年代や70年代のNASAにはみられる。

科学技術計算が大規模機械システムの作製に必須であることから、品質の高い数値計算用ソフトウエアに要求される条件を明確にするために、1970年米国科学財団ＮＳＦ（National Science Fondation）によってNATSプロジェクトがはじまった。米国内の主要大学と国立研究所が中心となってプロジェクトをすすめ、プロジェクトの決定にしたがって固有値計算や連立一次方程式などの計算ライブラリが作成された。数値計算ライブラリに必要な、計算精度、移植性、統一性、安全性、検査充分性、文書化、といった属性は現在でも充分に通じる考え方である。むしろ、必須であるにも関わらず、現在のほとんどのソフトウエアに欠けているものである。さらに、すぐれた数学ソフトウエアの開発には、数値解析の専門家、応用ソフトウエアの専門家、実施検査者である利用者、および全体を統括する委員会の４者が協力し合わなくてはならないことも示した。NATSに似たプロジェクトは英国でも同時期にNAG計画として遂行された。アメリカにつぐコンピュータ利用大国のわが国で一度もこのような試みがなされなかったところにソフトウエアに対する認識のなさが認められよう。

話はそれるが、1970年代の米国は医療に関係した研究が中心となり、科学技術計算用ソフトウエアの研究はややペースを落としたきらいがある。もっとも私立

図2.1

NASAのプロジェクトの考え方



長友信人「研究開発とプロジェクト管理」 より

１．３　数学ソフトウェアの品質

ＮＡＴＳプロジェクト
(National Activity to Test Software)

1970年に米国ＮＳＦ(National Science Foundation)の補助金により，ソフトウェアがもつべき品質特性を明らかにするとともに、良質なソフトウェアの作り方に指針を与えた。

・ソフトウェアの品質

(1)信頼性：十分な解析検討に基づいた良いアルゴリズムの採用

(2)頑健さ：誤り異常状態に対して十分配慮されており、アルゴリズムが作動すべき問題領域を明確にしている

(3)構　造：ライブラリ全体としてのまとまりと、世の中の標準化の動きに適合している

(4)使いやすさ：個々のサブルーチンは当然のこと、使用説明書(マニュアル)の類が利用者の技術水準に応じて整備される

(5)正しさ：各サブルーチンがどのような検証データで検査されているかを明らかにする

・このような品質をもつライブラリの設計目標

(1) 数学的にしっかりしていること

(2) 普遍的な開発言語を使っていること

(3) 検査が簡単で、誤りが少なく、使用説明書がしっかりしていること

(4) 種々な範疇の問題が解けるような解法の集まりであること

(5) 品質に対する測度があること

(6) コンピュータ社会における標準規格にのっとっていること

・数学ソフトウェアの開発には、

(1) アルゴリズムを提供する数値解析の専門家

(2) 各種のハードウェアに合ったルーチンを開発する応用ソフトウェアの専門家

(3) 特定ハードウェア上で実施検査(field test)をする利用者

(4) 全体計画を立案・統制する委員会

の四者が密接に連絡して協業しなければならない

ＮＡＴＳの場合には２４ケ所の大学・研究所が実施検査を手伝い、誤りや使用上の不備な点を指摘した。

・移植性

ある機種のコンピュータ用に開発されたサブルーチンやプログラムが他の機種のコンピュータにのる度合いをportabilityまたはtransportabilityという。移植性にもいろいろなレベルがある。

レベル１：プログラムを修正せずに、そのまま他機種に移せる

レベル２：ハードウェアの精度の違いや、コンパイラの違いによる手当てをするだけで移せる。これは手順が機械的で、utility programでできる場合に限る。

レベル３：オペレーティングシステムのデータ管理や領域管理の方式による違いを修正して移す。この場合はオペレーティングシステムに依存する部分を局所化しておかないと一般に大修正を要する。

大学の教官によるソフトウエア・ハウスの設立、ハードウエアとソフトウエアの価格分離、および応用分野の拡大という追い風に乗ってソフトウエア・ハウスがそれに代わって台頭するようになってきた。このため、使いやすさという観点からの図形処理との一体化が中心となり、計算中心から科学技術計算システムをめざす気配がでてきている。つまり、計算、図形処理、工学データベース、使用上のノウ・ハウ、といったワンセットでものをみる観点が浸透してきた。この小冊子はその延長上にある考え方を説明しているわけである。ＭＩＴの時代から４半世紀近くたった現在からあとは、コンピュータと通信と制御が深く結びついた処理に科学技術計算分野も向かうことになろう。

工学におけるコンピュータの底流を図に示したものが、図2.2である。1950年代のわずかなメモリしかないコンピュータでは単発の数値計算しかできなかった。それでも行列計算や数値積分、それにデータ構造やソーティングなどの基本的なものはこの時代にかなり研究されていて、Knuthが体系化して出版している。60年代にはいると、コンピュータもトランジスタの採用により高速化し、統計解析、線形計画法、有限要素法、回路解析、数値制御といった単一分野のかなりの規模の計算ができるようになった。さきに述べた夢の時代はこのような発展期のたまものであった。さらに70年代になると、データベースや図形処理装置が使用されるようになり、ＣＡＤやＣＡＭが普及することになる。そして80年代に入ると、スーパーコンピュータ、ワークステーション、ネットワークなどがつぎつぎと開発され、ハードウエアの制約がますます緩やかになってきた。図2.2からは、科学技術計算システムがデータ処理と結びついた工学情報システムのサブシステムとして構成される方向にあることが分かる。大学・研究所では科学技術計算システムという形で独立に存在できるが、企業の中では、業務上のデータ処理の中に組み込まれるか、接続された形で運用されることになる。この面を強調したのが、図2.3である。設計上の制約として仕様の個別化・多様化、納期、および費用が重きをおいてくれば当然のこととなる。

いま一つ忘れてはならないのは、強力なパソコンを個人が購入し、代表的な応用ソフトウェアやコンパイラがその上で動くようになったため、90年代は個人の時代の幕開けにもなるということである。科学技術計算の普及にとってこれは望ましいことで、メーカ主導型から解き放たれることになろう。

図2.2

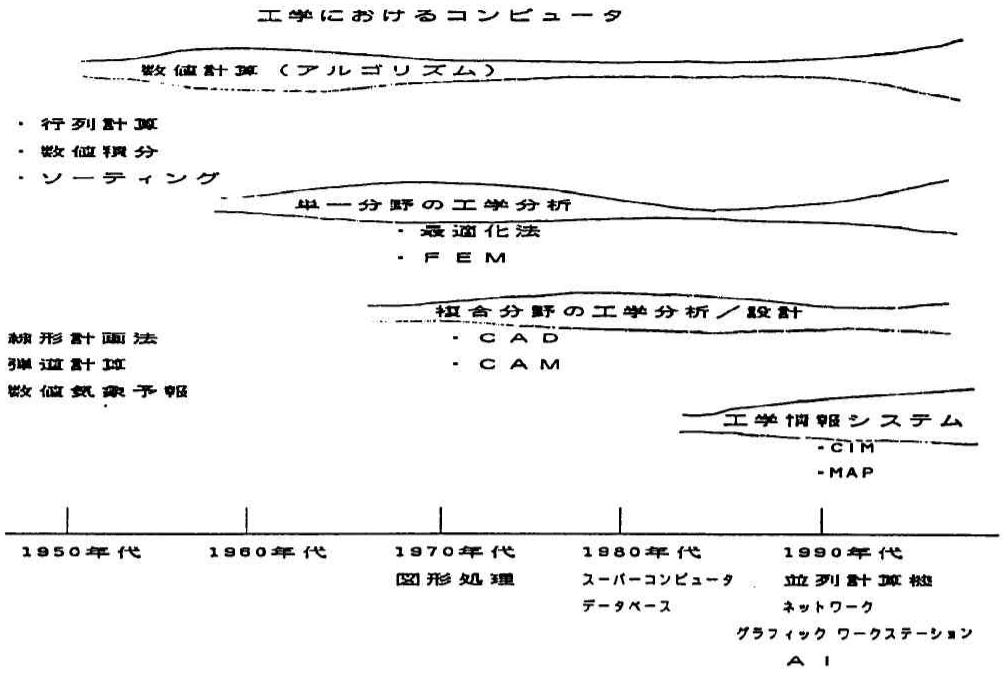


図2.3

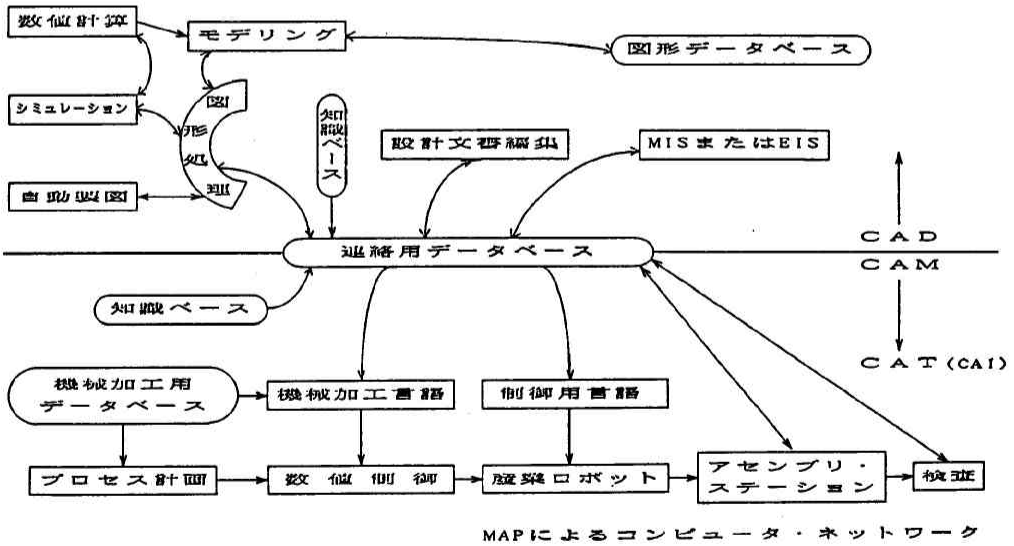


図 機械系ＣＡＥの全体像

3．最近のコンピュータ技術の発展

ここ10年間のコンピュータとそれに付随する数値解析法をごく簡単にふれる。1章でふれたように、コンピュータに関係するのは離散化法、数値解法、プログラム、および評価である。とくに、スーパーコンピュータのような特殊なハードウェアになると、その関係は大変に複雑となる。普通の汎用コンピュータでも加算・乗算命令にくらべ除算・判定分岐命令は数倍遅いが、スーパーコンピュータではハードウェア命令間の性能が10倍以上異なるし、それがマトリクスの規模にも依存して大幅に変わるというのだから、なおさら面倒である。また、10MIPSをこえるパソコンやワークステーションが身近で使えるようになりつつあるので、これも考えておかなくてはいけない。

科学技術計算用ソフトウェアが特定のハードウェア上で走るとき、その性能にはつぎの要素が関係してくる。

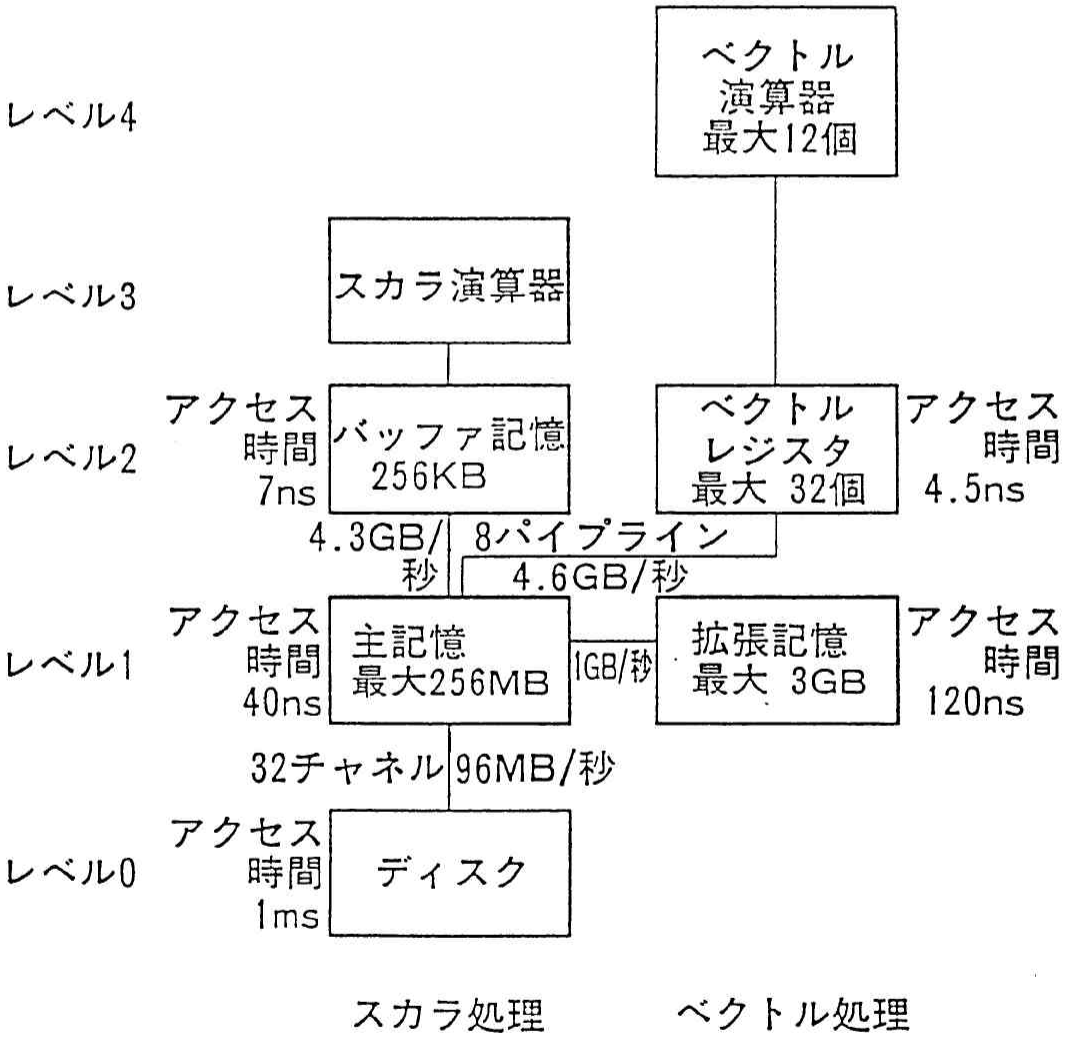
1．標準ハードウェア（制御装置、主記憶、キャッシュ記憶、ディスクなど）
2．特殊ハードウェア（ベクトル演算器、高速演算装置、拡張記憶など）
3．基本ソフトウェア（データ管理など）
4．Fortranコンパイラ（最適化の水準、データ配列方式など）
5．技術計算用ソフトウェア（プログラム技法、数値解法など）

たとえば、S-810スーパーコンピュータの場合、記憶装置を性能の面から階層化すると図3.1のとおりになる。日立製のM-680のような並列計算機ではさらに複雑な構成になり、図3.2のように記憶階層が一つ多くなり、利用者が性能を高めることがいっそう難しくなる。バッファ記憶にしても大容量作業記憶装置にしてもFortranの言語仕様になく、利用者が明示的に使えるとは思えない。しかし、数年後の国産スーパーコンピュータはこうした構成になるだろう。主記憶にしても高速にバッファ記憶にデータを取り込むために、バンクといういくつかの塊に分けられており、これらバンクは並列にデータを送り出せる仕掛になっている。連続した主記憶の番地はつぎつぎと別なバンクに割り付けられているので、連続した番地を扱うアルゴリズムが性能的によくなる。

ベクトル計算機がスーパーコンピュータとよばれるのは、配列（ベクトル）の高速処理に基づいている。これを内積計算の場合に図3.3により簡単に説明しよう。

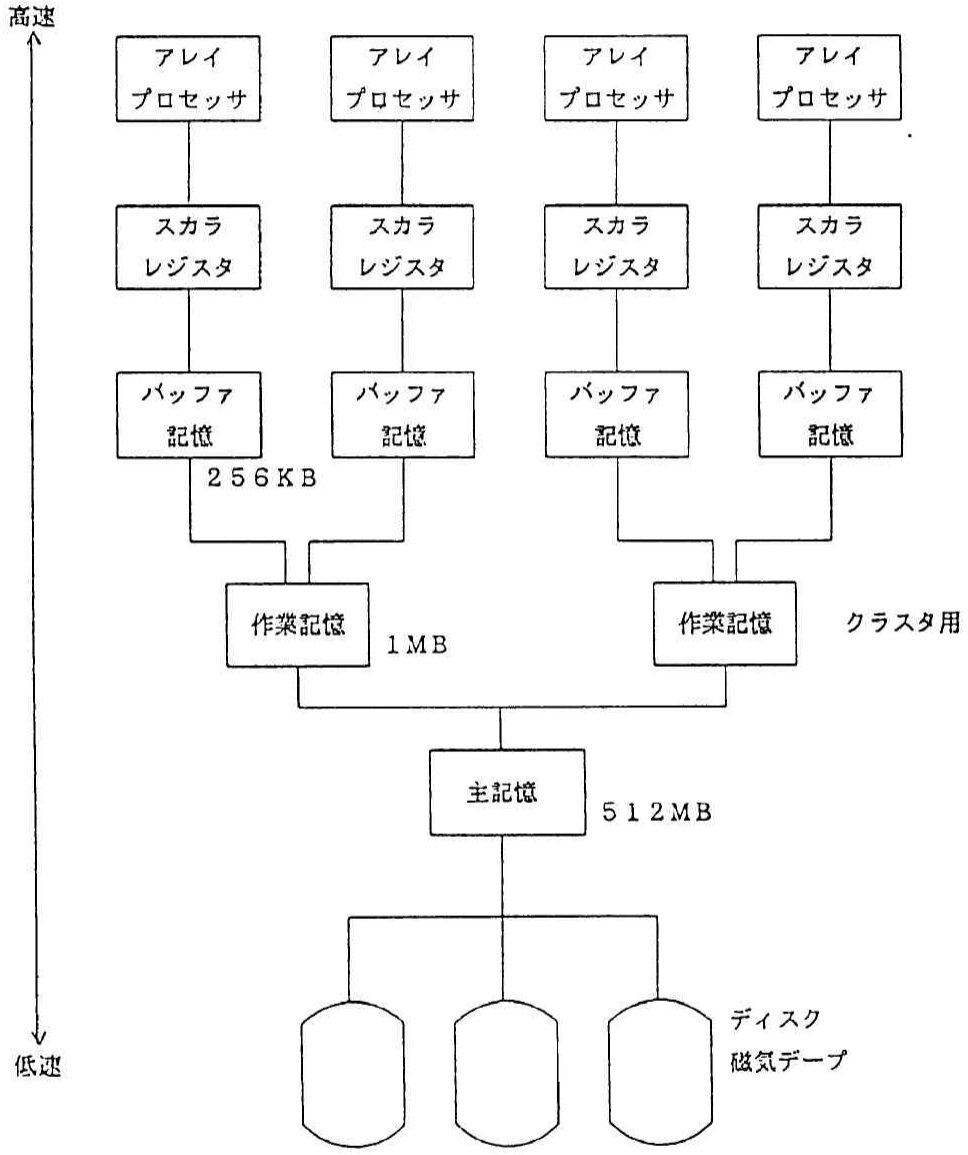
S-810のハードウェア

図3.1



パラレルコンピュータ　　図3.2

Ｍ－６８Ｘの場合



コンピュータにより配列計算をするとき、四則演算だけでなく、添え字の足し込み、添え字が配列の要素数に一致したかどうかの判定、一致してないときの反復計算のための分岐が必要となる。この判定や分岐はオーバーヘッドとなるが、ハードウエア命令の先取りが阻害されて本来の内積計算に要する時間にくらべ無視できない大きな時間を必要とする。しかも時間はどの程度離れたところへ分岐するかにも関係する。スーパーコンピュータの場合はこのオーバーヘッドをベクトルプロセッサとよぶ特殊なハードウエアによりなくして高速化をはかっている。もっとも、この特殊ハードウエアを立ち上げるために多少の時間を要するので、配列の要素数が少ないときにはスーパーコンピュータの利点はなくなる。

さらに、ベクトル計算機の中には演算器を複数個もつものがあり、図3.4に示すような二重ループの計算を複数の演算器を並列に使って高速処理ができる。つまり、演算器が複数個あるときには、ループ番号10のなかの二つの文は並行に処理できることになる。ここでの並列性は並列計算機でのＣＰＵの並列性とは違うことに注意すべきである。ベクトル計算機では、主記憶とベクトルレジスタ（バッファ記憶に相当する）があり、パイプラインは主記憶からの下り線と主記憶への上り線とに分かれていて、その本数も下りと上りとでは数が違う。普通は、下り線の本数が多く、一部の本数は下りと上りが共通に使えるようになっている。ベクトル計算機で性能を上げるにはこれらすべての本数にデータが等しく流れる必要があり、数式でいうと多項式になっていなければならない。図3.4でいうと下り車線は４本、上り車線は２本、しめて全部で６本あれば最も効率よくハードウエアは動くことになる。これ以上の車線があれば遊んでしまうし、これより車線が少なければ混み合って一部待ちを生じることになる。汎用コンピュータの場合も含めて、プログラムを実行するときに効率的なハードウエア利用の仕方を表3.1に示してある。ベクトル計算機や並列計算機をうまく使いこなすには競走車のレーサと同じように訓練と技術を要することになる。

このように、同一の数値計算法でもどのハードウエアを利用するかで、さまざまな工夫の仕方が考えられ、その有効利用にはノウ・ハウが必要である。この間の事情を図3.5にもとづいて説明しよう。図3.5は三好俊郎東大教授がS-810で離散化法と数値計算法との関連を調べた結果と、われわれが市販のプログラムを外部記憶装置としてディスクと拡張記憶を用いた結果、とを示している。ケース１の

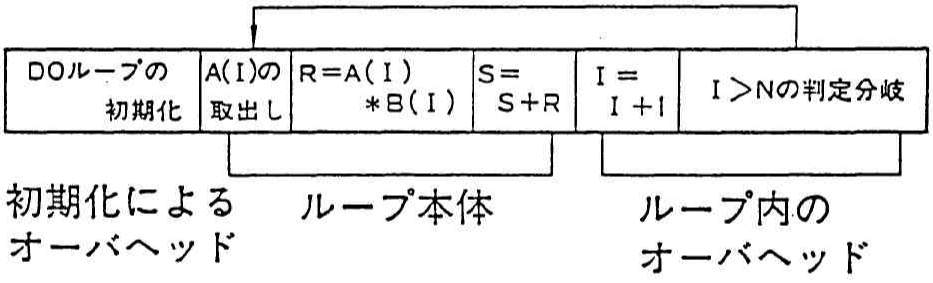
$S=\sum_{i=1}^{n} a_i b_i$の処理

図3.3

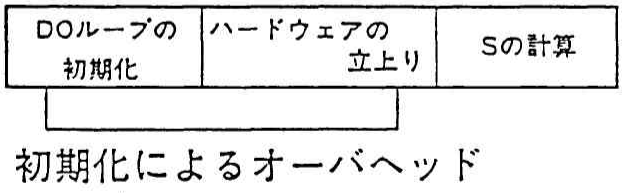
(1) Fortran 言語

```
      S=0.
      DO 10 I=1,N
10    S=S+A(I)*B(I)
```

(2) スカラ処理



(3) ベクトル処理



並列演算器

図3.4

```
      DO  20  J=1, NY
      DO  10  I=1, NX
10    Y(I, J)=Y(I, J)+D*X(I, J)
20    CONTINUE
```

⇩

```
      N2=NY/2*2
      DO  20  J=1, N2, 2
      DO  10  I=1, NX
      Y(I, J)=Y(I, J)+D*X(I, J)
10    Y(I, J+1)=Y(I, J+1)+D*X(I, J+1)
20    CONTINUE
      DO  40  J=N2+1, NY
      DO  30  I=1, NX
30    Y(I, J)=Y(I, J)+D*X(I, J)
40    CONTINUE
```

表3.1

効率的なハードウェア利用

１．ディスクと主記憶間
（1）ブロック入出力　（利用者）
（2）非同期入出力　（利用者）
（3）パラレル入出力　（FORTRAN）
（4）仮想入出力　（OS）
（5）仮想記憶装置　（OS）
２．主記憶とバッファ記憶間
（1）ブロック入出力
３．主記憶とベクトルレジスタ
（1）パイプラインの並列化（lordとstore）
（2）バンクコンフリクトの防止
４．ベクトルレジスタとベクトル演算器
（1）ベクトル演算器の並列化
（2）スカラ演算とベクトル演算の並列化
⇒Fortranコンパイラの最適化
（3）スカラ演算のベクトル演算化
⇒DOループの分割，DOループの入れ替え
（4）ベクトル長の増加⇒リストベクトル使用
５．拡張記憶
（1）ブロック入出力（MB単位）
（2）作業用領域
６．並列計算
（1）ループ内
（2）プログラム内サブタスク間
（3）プログラム間

モデルでは、有限要素としてアイソパラメトリック要素を用い、ケース２では有限要素として単純な線形要素を用いている。ケース１を有限要素法で離散化して得られる行列の一行あたりの非ゼロ要素数はケース２にくらべてかなり多くなる。したがって、ケース１では直接法が有利となり、ケース２では反復法が有利となることが予想され、結果はそのとおり２倍以上の開きがある。ケース１でオーダリング法を採用すれば、なお一層直接法が有利となろう。

拡張記憶は現在ベクトル計算機とＩＢＭの汎用計算機に装備されている外部記憶装置であり、経過時間を縮める点では圧倒的な利点を持っている。たとえば、図3.5のプログラムＢの場合には、結果を得るのに丸一日を要していたものが、わずか14分で結果を得ることができた。しかも課金に影響するＣＰＵ時間も30％以上も減っている。これはディスク入出力では、入出力チェックやバッファリング（データをまとめて管理して入出力時間を少なくするやり方）にかなりのＣＰＵ処理が要ったのに対し、拡張記憶ではこれらの処理が不要になったためである。しかし、計算センタのようなマルチジョブによる運営の時にはほかの利用者に対して迷惑となるかもしれない。

もう一つの例を上げよう。図3.6は、オランダのVan der Vorst教授が京大に客員教授として滞在していたときに諸種のスーパーコンで同じプログラムを実測した結果である。ハードウェアごとに解法間の違いが異なるとともに、問題の規模が変わると、解法やコンピュータの優劣が変化していくのがよく分かる。しかし、メーカのマニュアルを読んだだけでは結果は予想できないし、解釈もできない。小、中規模のデータではスケーリングつきのＣＧ法がいずれのコンピュータでも一番速いが、大規模になると、オーダリングつきのMICCG法が最も速くなる。ICCG法のベクトル版もCRAYでは効果がないが、ＮＥＣや日立のスーパーコンピュータでは非常な効果が出ている。さらに、各種のコンピュータを同一の数値計算用サブルーチンを実行させたとき、どのようになるかを図3.7に示す。公称の最高（ピーク）性能と実行性能がどんなに違うかがよく分かる。スーパーコンピュータ用にプログラムを改善しても、100X100程度の行列ではわずか10％にも満たない性能しか出ていない。これに対してミニスーパーコンピュータとよぶ機種はかなり健闘しているといえる。

ここ数年のコンピュータ、端末装置、伝送装置の進歩は、要求にはまだ満たな

図3.5

処理の全体（構造解析の場合）

- 離散化法と数値解析

  ケース１：３次元表面き裂の進展解析
  （有限要素としてはアイソパラメトリック要素）

  ケース２：直方体の単軸引張り解析
  （有限要素としては線形要素）

有限要素法（ICCG法）		有限要素法（スカイライン法）		境界要素法（ガウス法）
ケース1	ケース2	ケース1	ケース2	ケース1
26.1	2.6	7.3	5.3	6.2

単位：秒

- 拡張記憶とディスクの差

プログラムA				プログラムB			
ディスク		拡張記憶		ディスク		拡張記憶	
CPU	経過	CPU	経過	CPU	経過	CPU	経過
S	S	S	S	M S	H M	M S	M S
282	2518	260	360	18 52	7 31	1329	14 7

図3.6

各種反復法の比較(Henk A. van der Vorst)

・モデル1 ： $-u''_{xx} - u''_{yy} = 0$　　$n_x = n_y = 30$,　$N = 900$

手　法	回	CRAY X-MP	VP-200	S-810/20	SX-2
ＩＣＣＧ	２１	０．０１３	０．０２３	０．０２２	０．００９
ＭＩＣＣＧ	１６	０．００９	０．０１８	０．０１６	０．００７
ＩＣＣＧ（順序付け）	２１	０．０１０	０．００９	０．０１０	０．００４
Ｓｃａｌｅｄ　ＣＧ	６３	０．００９	０．００６	０．００７	０．００３
ベクトル化　ＩＣＣＧ	２３	０．００８	０．０１１	０．０１２	０．００５

単位：ＣＰＵ秒

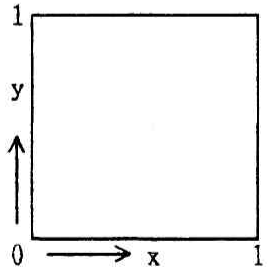
・モデル2 ： $-(D u'_x)'_x - (D u'_y)'_y = f$

$n_x = n_y = 100$,　　$N = 10,000$

手　法	反復回	CRAY X-MP	VP-200	S-810/20	SX-2
ＩＣＣＧ	１２９	０．６４２	１．３４	０．８３９	０．３６０
ＭＩＣＣＧ	１４７	０．７４９	１．５４	０．９４９	０．４０８
ＩＣＣＧ（順序付け）	１２９	０．４３３	０．２１７	０．２６４	０．１０６
Ｓｃａｌｅｄ　ＣＧ	３４２	０．５０５	０．２０７	０．２６０	０．０９６
ベクトル化　ＩＣＣＧ	１３７	０．４４０	０．３０９	０．３４７	０．１２１

単位：ＣＰＵ秒

モデル1

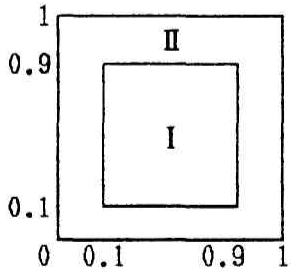


境界条件：ディリクレ条件

$u(x,0) = u(x,1) = \sin((n+1)x)$

$u(0,y) = u(1,y) = \sin((n+1)y)$

モデル2



Ⅰ：　D＝１０００

　　f＝１０００

Ⅱ：　D＝１

　　f＝０

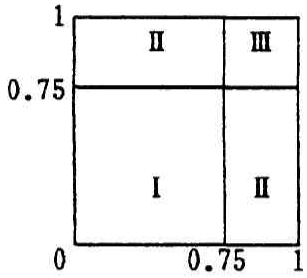
境界条件：ディリクレ条件　$u(x,0) = 0$

他の境界で　$\partial u/\partial n = 0$

各種反復法の比較(Henk A. van der Vorst)

モデル3 : $-(Du'_x)'_x - (Du'_y)'_y = f$

$n_x = n_y = 200, \quad N = 40,000$



I : $D = 10^5$ $f = 1$

II : $D = 1$ $f = 1$

III : $D = 0.1$ $f = 1$

境界条件：ディリクレ条件，すべての境界で u = 0

手　法	反復回	CRAY X-MP	VP-200	S-810/20	SX-2
ICCG	132	5.09	2.51	5.30	1.30
MICCG	54	2.07	0.987	2.17	0.534
ICCG (順序付け)	132	3.04	1.42	0.693	0.299
MICCG(順序付け)	54	1.25	0.582	0.283	0.122
Scaled CG	437	−	2.22	0.996	0.461
ベクトル化 ICCG	145	3.77	1.63	0.963	0.374

単位：CPU秒

LINKPACK ベンチマーク 100×100の連立一次方程式

図3.7

計算機	ピーク性能	修正前		修正後	
	MFLOPS	MFLOPS	比	MFLOPS	比
Culler PSC	5	2	.40		
Multiflow TRACE	15	6	.40		
CONVEX C-1	20	3	.15	9.6	.48
SCS-40	44	8	.18	13	.30
FPS-264	54	5.6	.10	24	.44
Alliamt FX/8	94(8CPU)	7.6	.08	11	.11
Amdahl 500	133	14	.11		
CRAY-1	160	12	.0.75	38	.24
CRAY X-MP-1	210	24	.11	48	.23
I3090/VF200	216	12 *	.11	24 *	.22
Amdahl 1100	267	16	.060	48	.18
NEC SX-1E	325	35	.11	71	.22
CDC CYBER 205	400	17	.043	24	.06
CRAY X-MP-2	420	24 *	.11	37 **	.088
I3090/VF-400	432	12 *	.11	24 *	.22
Amdahl1200	533	18	.034	52	.098
NEC SX-1	650	39	.06	74	.11
CRAY X-MP-4	840	24 *	.11	74(4CPU)	.088
H S-810/20	840	17	.020	48	.057
NEC SX-2	1300	46	.035	100	.076
CRAY-2	2000	15 *	.030	28 *	.056

Jack J.Dongarra (Argonne研究所)

いとはいえ、目を見張るものがある。たとえば、ＩＢＭのGomory博士が1985年に講演したことによると、1990年代初頭のハードウェアの性能は表３．２のとおりになるという。パソコンは１MBの主記憶をもつ32ビットパソコンンが主流となったし、その先のスーパーパソコンも試作段階にある。しかも、大容量のディスクや記憶装置も使用できるようになる。これら高性能なパソコンが個人で購入できることにより、科学技術計算用ソフトウェアの試作品や構成部品の開発・テストは飛躍的に効率を上げることができるだろうし、中小企業による応用ソフトウェア製品の生産がうまくいくようになるだろう。本来、応用ソフトウェアは中小企業が担当すべき分野なのである。

スーパーコンにいたってはベクトル計算機、並列計算機（マルチプロセッサ）、超並列計算機（パラレル・プロセッサ）、専用コンピュータ、などさまざまなコンピュータが市販されつつある。また、値段的にも、ミニスーパーとかマイクロスーパーといった廉価版も普及のきざしをみせている。なかでも、並列計算機は、ＩＢＭが1988年から本格的に市場に参入してきたことにより、競争も激化している。これまでの商用の科学技術計算用コンピュータの経過をみると、図3.8のとおりになる。ＤＥＣ社のスーパーミニコンピュータからはじまって、汎用機＋ベクトルプロセッサ、パイプライン型ベクトル計算機、並列計算機＋ベクトルプロセッサ、多重パイプライン型ベクトル計算機、など、高速化をはかるために実にさまざまな工夫がなされてきた。当然、コンピュータは複雑になり、それを使いこなして性能を出すことは至難の技となってきた。しかも、ハードウェアの設計者もオペレーティング・システムやコンパイラの設計者も、利用者側に立ってハードウェアのピーク性能を実現するやり方を知らないのである。

ハードウェアの開発技術・製造技術はコンピュータの導入により年々改善され、市場の需要も増えるにしたがい、ハードウェアの製造原価はどんどん低下している。しかし、上に述べたような事情により、ハードウェアをうまく使いこなす科学技術計算用ソフトウェアを作成するのは年々難しく、しかも効果が相対的に小さくなりつつある。図3.7に示したLINPACKルーチンの場合は極端であるとしても、いったんでき上がっているソフトウェアをわずかなプログラム修正で大きな改善効果をあげることはあまりできないといえる。しかも、同じように見えるハードウェア・アーキテクチャをもつコンピュータといえどあまり信用はできない。IB

表3.2

コンピュータの動向 IBM社研究担当副社長
GOMORY博士

1．パーソナルコンピュータ
1988年 5～10MIPS　1991年 10～20MIPS

2．ワークステーション
（1）標準タイプ
10MIPS　1～16MBの記憶容量
高速、高精度カラープリンタ
（2）高級タイプ
20MIPS　16MB以上の記憶容量
マルチプロセッサ
（3）自然語機能　1993年
手書き入力
10K～100K語の辞書
20～30MIPS
ポータブル
ホストコンピュータのデータベース
のアクセス

3．大規模システム
1000MIPS

4．ミニコンピュータ　1993年
35MIPS
マルチプロセッサ
部門毎のワークステーションとなる

5．ディスクの記憶密度

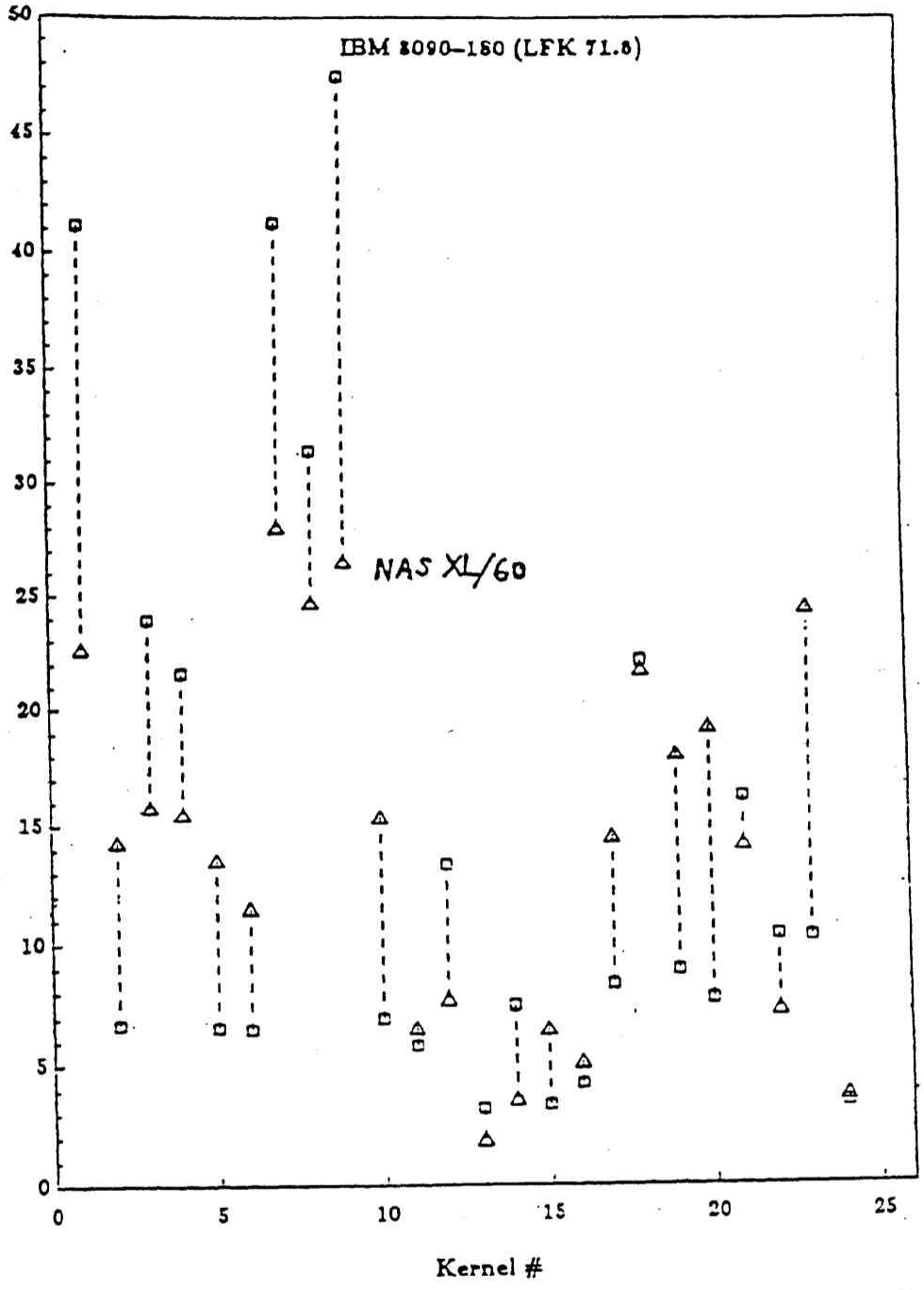
1980年	1985年	1990年	1995年
1	2	6	10～40

M3090-180とNAS-XL/60（日立のM-280H）とは同じようなアーキテクチャをもち、両者ともベクトルプロセッサをもつ。この似たようなコンピュータにリバモア研究所で用意したリボモア・ループという、23個の異なった分野におけるプログラムからなる標準的なベンチマーク・テストを実行したところ、図3.9のとおりになり、勝ったり負けたりで甚だしい性能の違いが生じた。ほとんどのプログラムで２倍もの性能の違いが出ている。しかし、平均すれば引き分けである。特定の分野の計算をする人にとっては「平均」の話では困る。コンピュータ・ネットワークにつながるコンピュータの中から、いちばん安く速く計算できるコンピュータを選びたい。このためには、コンピュータ・メーカ側から性能データベースが提示される必要がある。しかも、分野ごとの代表的なプログラムに関する実測データを公開しなくてはいけない。もちろん、自分たちだけでできないだろうから、国内外の大学や研究機関の専門家の指導を受けることになろう。ハードウェア・メーカはどうしても製品に関する情報を閉ざしがちであるが、ネットワークにつなげる以上は情報の最低限の公開はせざるをえないだろう。コンピュータ関連業界は情報機器を販売しているにもかかわらず、情報に対する理念といったものに欠けている。知的な財産とその内容がどういうものかについて日米では認識の差があまりにも大きいようだ。

スーパーコンピュータの歴史　　図3.8

1 MFLOPS	スーパーミニコン（VAX8600）
10 MFLOPS	汎用機(M680, ACOS1500)
	汎用機＋付加VP(IAP, FPS, VF)
	ミニスーパーコン(CONVEX)
100 MFLOPS	パイプライン型(CRAY1, CYBER205, S810, VP)
	汎用機＋多付加VP(IBM+10FPS)
	ミニスーパーコン(FX/8 8CPU, FPS MODEL1600)
500 MFLOPS	パイプライン型(S810, VP, SX)
	多重パイプライン型(CRAY-MP,CRAY2)
	汎用機＋多付加MP(RP3 512CPU, FPS-T 10000)
1 GFLOPS	パイプライン型(S820, VP2000, SX-2)
	多重パイプライン型(CRAY-MP)
	NASAプロジェクトNASF
5 GFLOPS	多重パイプライン型(ETA10 8CPU)
10 GFLOPS	専用機(GF11:QCD 566CPU)
	多重パイプライン型(SX-3 4CPU, CRAY3 16CPU)
	通産省プロジェクト
100 GFLOPS	
	多重パイプライン型(CRAY4 64CPU)
1 TFLOPS	（次世代の目標）

図3.9



LLNL Ftn Kernels (DOspan=471)

4．これからの科学技術計算システム

各種のスーパーコンピュータが普及し、ＩＢＭが本格参入するにしたがって、技術計算分野は大きな刺激をうけている。このあたりの事情をＩＢＭのWeis副社長とボーイング・コンピュータ・サービス社のErisman氏の公演内容をそれぞれ表4.1と表4.2に示す。両氏とも、これからの科学技術計算が

1．図形処理装置をもつ高性能なワークステーション（端末中心の考え）
2．並列計算
3．ネットワーク
4．コンパイラ技術
5．データ管理
6．アルゴリズム
7．異種の計算の統合
8．ＡＩ

を道具として、より困難な分野の計算をこなすようになろうと述べている。また、物理現象の解析にあたって必要なテラFLOPSの計算も10年先には手が届く範囲になるだろうとみている。さらに、これからの科学技術計算システムの開発に当たっては、NATSプロジェクトで明らかになった各専門家の相互理解とともに、アルゴリズムの特許化という企業論理にいかに対処していくかが課題になる。

4．1　ワークステーションとパソコン

自宅や職場での32ビット・パソコンの普及はめざましいものがある。とくに1988年にマイクロソフト社が最適化Fortran77のV.4を発売してから、大型汎用計算機とほぼ同じ言語仕様のFortranプログラムが作成できるようになった。しかも、パソコンで作成したフロッピ・ディスクのソース・プログラムは汎用機のMS-DOS用ワークステーションを通じて汎用機でも処理できる。一方、高精度の図形処理装置やレーザビーム・プリンタを接続することによってモデルの作成や結果の編集出力が図形のかたちで実行できる。しかも、計算の途中過程を図形表示できれば、ムダな計算がつづくのを防ぐことができる。ただし、図形処理についてはパソコンではクローズ型のCAD(computr　aided　design)システムが中心となってい

表4.1

スーパーコンピュータの大挑戦　　Alan Weis IBM副社長

・スーパーコンピューティングはいま「百花斉放」
・使うとき、経験、使いにくさ、試行、「スーパー」利用者に限定してはだめ。
　費用効果、信頼性、使いやすさ、ソフトウェアの豊富さ（治工具と応用）、
　頑健なデータ管理
・ユーザは政府・大学・産業の研究者から、産業・ビジネスのエンド・ユーザへ
　(1) 科学工学における基本問題への適用・・・材料科学や超音速機
　(2) 諸産業にまたがったスーパーコンピュータ・システムや応用
　　高品質の製品を、短時間で、かつ少ない費用で開発することがその例
・広帯域入出力が必要：　データ、視覚化、ネットワーク・アクセス
・バランスが大切で、その構成要素は
　(1)性能（ベクトル、スカラ）　(2)並列性　(3)大規模記憶
　(4)広帯域入出力サブシステム
・ソフトウェアとして
　(1)対話計算（大規模仮想記憶）・・・テラバイトの実記憶の時代
　(2)制御システム（仮想プロセッサ、物理プロセッサ）
　(3)コンパイラ技術（ベクトル化、並列化）
　(4)データ管理（信頼性、アクセス）
　(5)ネットワーク結合
　(6)ワークステーションの統合と結合
・視覚化として
　(1)スーパーコンピュータのデータを処理できる
　(2)計算を「のぞき見る」
　(3)高解度、広帯域のワークステーション・・・70MB/秒
　(4)ジョイスティックなどを使った応用
・ネットワーク・アクセスとインタコネクション
　(1)高帯域　(2)サービス管理（プログラムとデータ）
　(3)OSI標準への移行　(4)NSFNETネットワーク
・アルゴリズムの拡張と応用の展開
　(1)計算化学　(2)金融界の意思決定
　(3)製造・プロセスの制御　(4)統合CAD/CAE/CAM
・処理を千倍に
　CPUの性能を10倍に、並列度により10倍に、視覚化により10倍に、しめて千倍
・IBMもついに真面目に商売を始めた
　仮想プロセッサ、並列Fortran

表4.2

製品開発の道具としてのスーパーコンピュータ

A.M.Erisman (Boeing Computer Services)

・科学計算の世界　：Fortran、スーパーコン、ミニコン、ワークステーション

・データ処理の世界：第４世代言語、メインフレーム、パソコン

・新しい製造の動き：製品を作るとき設計条件を課し、分析すべきモデルを変える。
低価格で製品を作るために、分析代替案は費用データが必要

・分析とCAD/CAMとの堅い結びつきが必要
スーパーコンの利用者はCAD/CAMのことについて知らないし、その逆もいえる

・乗り物との比喩　：

歩く	一人のユーザ	パソコン
自動車	一人のユーザ	ワークステーション
ジェット機	マルチユーザ／エンジン	ベクタスーパーコン
宇宙旅行	hostile環境	明日のスーパーコン

・成功ストーリ
科学計算（アルゴリズム、応用、環境〔データ管理、ネットワーク、ＣＧ〕）
計算機科学（ＯＳ、コンパイラ、言語、・・・）
研究と製品開発との間のリンク、エンドユーザによる利用が必要

・もっと使うための潜在力

(1)設計最適化

(2)統合解析
・データの交換－－構造解析、制御系、熱解析、空力学の間
・ＡＩ／記号計算の導入
・設計に製造のことを入れる(tradeoff)
…advanced optimization expert system
・全体プロセスのモデリング

・相互理解

(1) アーキテクチャ屋とアルゴリズム屋・・・ベクタマシン
ライブラリの大切さ‥‥移行性と性能

(2) アーキテクチャ屋、アルゴリズム屋、応用屋・・・並列コンピュータ

・統合の必要性

(1) スーパーコン、ワークステーション、グラフィック、データ管理、
ＡＩツール、広域コミュニケーション、共通ＯＳ

(2) 端末ユーザが中心でスーパーコンはバック

・要約

(1) モデルの視覚化(steer simulation)

(2) モデリングとアルゴリズムの開発

(3) 統合化した計算システム・・・スーパーコンとワークステーション

(4) データ管理と会社組織

(5) 経営に対する利点の確立

て、互換性のあるサブルーチン型のソフトウェアがないために、汎用機にもっていくことはむずかしい。

パソコンの普及にともない、教育の形態や技術計算関係の専門書の形態も変化してきた。昨年末に丸善より出版された「Fortran77による数値計算ソフトウェア」のように、ソースプログラムをフロッピ・ディスクに入れ、読者が簡単にプログラムを使用できる時代になった。まだ日本では例外にすぎないが、このような出版の形態が進めば、専門書に説明用の図表がカラーで入った光ディスクがついてくることも夢ではない。あるいは、ディスクトップ・パブリッシング用に、本の代わりにフロッピ・ディスクが売られることにもなるだろう。これらのフロッピを教室や自宅のパソコンで共通に利用できるようになれば、コンピュータや科学技術計算の演習はきわめて身近なものとなり、しかも低価格のカラープリンタの登場により効果は絶大なものとなろう。もちろん、教師側の理解と努力も必要である。

4．2　並列計算

4.7に述べるように、これからは異種の計算を並行に処理することによって計算を進め、途中結果を含めて結果の可視化により全体の計算効率を高めることが必要となる。これら計算と可視化の過程で並列処理は新しい世界を開いてくれる。1946年のENIACは１秒間に千回の乗算ができたという。これに対し、85年の日本電気のSX-2は１秒間で10億回の演算ができる。つまり、40年間で百万倍となり、平均すると７年で10倍になる。しかし、最近の半導体素子の性能向上をみると、76年のCRAY1にくらべ85年のCRAY2は３倍にとかなっていないし、もうじき出荷となるCRAY3はわずかにその２倍にしかならない。89年に発表した世界最高速のSX-3にしてもCRAY2よりやや速いだけである。このように半導体素子の速さが限界に近づいてきたため、コンピュータの性能を上げるには、各種の並列化や階層化をはかって、それをソフトウェアでうまく制御することが必要となった。このため、米国ではすでに並列計算機の時代に入っており、日本も数年先にはそうなるはずである。

並列性と階層性の実現の仕方にはさまざまなやり方があり、各社それぞれの独自な方式を採用している。基本ソフトウェアやコンパイラも当然これらハードウ

スーパーコンピュータの性能（商用）　　表4.3

国名	機種	メーカ	発表	ピーク性能
	CRAY-1	CRAY	1976	80
	CRAY X-MP	CRAY	1982	400
	CRAY-2	CRAY	1981	800
	Cyber 205	CDC	1980	400
米国	FPS-164/MAX	FPS	1984	341
	X-MP/4(4CPU)	CRAY	1984	800
	ETA10(8CPU)	ETA	1986	10000
	Y-MP/832(8CPU)	CRAY	1988	4000
	CRAY-3(16CPU)	CRAY	1989?	16000
	CRAY-4(64CPU)	CRAY	1992?	128000
	S-810/20	日立	1982	630
	VP-200	富士通	1982	500
	SX-2	日電	1983	1300
日本	VP-400	富士通	1985	1142
	S-820/80	日立	1987	3000
	VP-2000(2CPU)	富士通	1988	4000
	SX-3(4CPU)	日電	1989	22000

1946年　ENIAC　10^3 FLOPS

1985年　SX-2　10^9 FLOPS = 10^3 MFLOPS = 1GFLOPS

40年で10^6倍　　7年で10倍（平均）

1976年　CRAY1　clock period = 12.5nsec

1985年　CRAY2　clock period = 4.1nsec

9年で3倍

1989年？ CRAY3　clock period = 2.0nsec

1992年？ CRAY4　clock period = 1.0nsec

？　ChenMP clock period = 1.0nsec

今後は5 - 10 年で5倍以上のclock periodは無理

並列化にならざるをえない

CRAY3は16CPU, ChenMPは64CPU, SX-3は4CPU

（注）	クロック・サイクル	スカラ性能
NEC SX-3	2.9ns	160MFLOPS
S-820	4.0	
CRAY Y-MP8/8	6.0	192

エアの特性に応じて開発されることになる。このため、利用者の作るソフトウェアはベクトル計算機とは較べようもないくらい、ハードウェアの固有の性質にひきづられることになり、アルゴリズムやプログラムの移植性が問題となろう。さらに、比較的少数のＣＰＵをもったマルチプロセッサ型の並列計算機と、多数のＣＰＵをもった超並列型の並列計算機ではプログラムのアルゴリズムも違ってくるだろう。マルチプロセッサ型ではプロセッサ間の処理の配分や同期化が技術的問題点となり、超並列型では多数のプロセッサへの双方向の放送が問題点となる。したがって、マルチプロセッサ型では従来路線の延長でもある程度いけるが、超並列型ではものの見方を替えなくてはならないだろう。

並列性によって、アルゴリズムやプログラムをうまく作って性能を上げることができたとしても、反対にソフトウェアの維持・管理・検証・安全性については従来と較べてはるかに面倒なものとなろう。コンピュータがシステムとしてより一層複雑になるため、原子力や航空機のように巨大システムが抱える問題点が科学技術計算システムについても当てはまることになる。超並列型は使用目的に応じてつぎのような専用のものがある。

（１）アニメーション用
（２）画像処理用
（３）ニューロコンピュータ
（４）セルラ・オートマタ・マシン
（５）データ・フロー・マシン
（６）有限差分法用
（７）量子化学用

4.4でふれるが、並列用言語やデバッグ用・性能評価用の治工具を整備して利用者に提供できるかどうかが重要なポイントになる。単に並列用Fortranだけではダメで、問題記述用の言語が必要となろう。この種の言語の開発には利用者としての現象屋、数値計算屋、言語屋、ハードウェア屋の協働が必要である。

４．３　コンピュータ・ネットワーク

高速度の技術計算を実現するにあたって、第一の本質的な問題は、ワークステーションとスーパーコンをつなぐネットワークの構築とその伝送速度である。米

表4.4

科学計算用ネットワーク

・動画像の伝送

白黒　　10Mbps ～ 60Mbps

カラー　329Mbps ～1920Mbps(*)

圧縮技術を使うと10～1000倍小さくできる

・数百台のワークステーションをスーパーコンに接続

４ＧＢのデータを数百Mbpsで送る必要がある

（現状）

今日のネットワークは1000倍遅い

NSFNETでは　56Kbps

→ 1988年７月に幹線は1.5Mbps

→ 1989年３月に45Mbps

・グラフィックス・スーパーコンピュータ

タイタン：4CPU，最大性能 64MFLOPS

62.5ns（CRAY X-MPは8.5ns）

1万3千ドル／MFLOPS

（CRAYは20万ドル）

1秒に20万ものフルカラーの多角形作成

数年先：性能がミニスーパー並み、(**)

価格はＷＳ並み

(*)1000×1000×24ﾋﾞｯﾄ×30HZ＝720Mbps

(**)Nelson Computer Research社のマイクロスーパーコン

Cray1の1/5～1/10の性能で15,000ドル

国の大学・研究所では国防総省のARPAネットワークと、各種の構内ネットワークとを接続しているが、これもまた、画像伝送に必要な希望速度にくらべ千倍おそい。このため、米国科学財団はARPAネットワークにかわるNSFNETネットワークの構築に向かっている。これは、コンピュータ資源を有効活用して科学技術計算の生産性を上げることを目指しており、米国のハイテク戦略の一環をなしている。伝送速度をあげるには、情報の圧縮・分担によって情報量を少なくするか、伝送路の容量を向上させるしかない。情報量を少なくするには、ワークステーション側が多容量・高性能にならざるを得ないし、数年先にはワークステーションもスーパーコンに近い性能をもつとみられている。当然、コンピュータ・ネットワークにおけるデータの標準化が必要となるし、暗号化などの安全性についても研究がなされている。

コンピュータ・ネットワークによってさまざまな業務がつながってくると、科学技術計算システムも全体システムの中の一要素にしかならなくなる。同時に、ネットワーク・サービスを売る企業にとってはこれまでの科学技術計算領域のほかに、金融・証券・保険・プロジェクト管理・経営計画といった計算領域も扱うことになり、理工系の技術者を必要とすることになる。

4．4　コンパイラ技術

各種のスーパーコンに見合った性能をもつアルゴリズムとプログラムの開発が問題である。コンパイラ技術が進歩するといっても、多様なハードウェアをどの程度支援できるかは非常に不透明であり、どうしても、科学技術計算用ソフトウェアの作成者にしわよせがきてしまう。しかし、現在のように数値シミュレーション用ソフトウェアを開発する専門職を意識的に育てようとしない場合、たいへん先行きが暗いといえよう。ベクトル計算機の場合性能を上げるには、

1．最内側のベクトル長を大きくする。
2．並列演算器を有効活用する。
3．load/store用のパイプラインを並列使用する。
4．拡張記憶をうまく用いる。
5．ベクトルプロセッサとスカラプロセッサを並行に動かす。
6．ベクトル化率を向上させる。

があるが、諸種のハードウエア資源を並列化することになり、ともすれば職人芸的なプログラム技法に走りかねない。

並列計算機の場合には性能を上げるための並列処理技法につぎの４とおりが考えられる。最初の二つはFortranコンパイラの問題になる。

１．ＤＯループ内で添え字ごとに異なるプロセッサを割り振る。

２．データフロー分析により、データ依存のないコーディング部分を別なプロセッサで並行に実行する。

３．Fortran言語レベルでタスクないしはジョブステップの制御を利用者自身がする。

４．ジョブ単位の並列処理をＯＳが面倒をみる（現在の多重ジョブがそれ）。

Fortran言語レベルのときも、ＣＲＡＹのようにサブルーチン・コールで処理する場合と、ＩＢＭのParallel Fortranのように実行文として含ませる場合とがある。さらに、並列に起こる物理現象を記述するのに便利な現象指向の並列言語も考えられる。このような並列言語はＣ言語やパラレルFortranを使って作成することになろう。また、並列用のプログラムはデバッグがけた外れに難しくなるので、そのための治工具が多種開発される必要がある。つまり、プログラム開発環境を大幅に整備しなくてはいけない。

４．５　データベースと知識ベース

ＣＡＤなどにおける設計情報や形状情報をデータベースに蓄え、設計作業に利用するだけでなく、設計文書の作成と、原価計算などの事務処理に接続させることができなくてはならない。しかもこのとき、設計者のノウ・ハウが知識ベースとして利用できることも望ましい。このような多目的なデータベースを構築し維持するには米国のGIDEP（Government-Industry Data Exchange Program）のように国家機関が指導して仕掛けをつくることが必要かもしれない。これまでデータベース・システムというと事務用のデータベース・システムが中心で工学的なデータベース・システムはあまり問題とされてこなかった。わずかに図形データベース・システムがＣＡＤの道具として個別に開発されてきたにすぎない。しかし、そろそろ科学技術計算システム用のデータベース・システムとデータベースの検討と構築が始まってもよい時期である。

ネットワークにつながる諸種のコンピュータを有効に使うための知識や、問題を部分構造に分けて精度よく並列計算向きにすることも計算効率を上げるノウ・ハウである。このような使用上の知識ベースもきわめて重要であり、利用者とメーカの協力が必要となる。いずれにしてもこのような知識ベースの作成には膨大な時間と費用がかかるので、その費用をハードウェア・メーカや利用者が負担するかという問題がある。しかし、これらの知識ベースの恩恵を受けるのは道具としてのコンピュータを売るメーカと、それを使う利用者と、国家なのである。

4．6　アルゴリズム

数値解法を含めてアルゴリズムの分野では並列化の研究が非常にさかんである。しかもこれが契機となって、従来の解法が見直され、新たな解法が発表されてもいる。わが国ではこの分野の認識が薄く専門家も少ないが、出版形態の変化により新しいアルゴリズムにしたがったプログラムがフロッピ・ディスクで簡単に使用できる方向にあるので、誰でもこれにふれる機会が増え、関心をもつ学生が出てくるとともに、教育の水準は少しずつ上がるかもしれない。

現在、ごく基本的な部品ルーチンはソフトウェア・ハウスやハードウェア・メーカにより市販されているが、これらはごく限られた基本領域に関するものであり、応用ソフトウェアを作成するにはあまりにも種類が少ない。たとえば、有限要素法や回路解析や数理計画法などのシステムを作ろうとしても、それに合った強力な部品集は見あたらない。市販されている部品ルーチンはデータ構造もごく単純で、外部記憶装置も利用せず、部品間の関係も乏しい。マイクロプロセッサのように標準化されていて、利用者がうまく組み合わせて自分の用途にまとめ上げるといったことがなかなかできない。こういった多目的の汎用部品ルーチンの市販が強く望まれる。

4．7　異種の計算の統合

計算能力が増大してきたことにより、いままでは部分構造ごとに分けて計算したり、熱、流体、機構、制御を別々に計算していたのが、統合して計算できる可能性がでてきた。このような例としては、自動車の内燃機関内の計算、航空機の設計計算、構造物の最適設計、およびロボット制御といつたものがある。内燃機

関では、熱源および力源、ピストン、クランクへの接続部があり、形状と材料に応じていくつかの偏微分方程式を同時に解くことが望ましい。また、最適設計では、有限要素法と最適化法を交互に解いて、ある制約条件に応じた構造物の設計をしたい。つまり分析とＣＡＤの結びつきが強くなる。さらに、ロボットのように関こう解析と電気制御とを組み合わせた解析が必要となる。動いているロボットの腕を急に止めたときにロボット自身に及ぼす負荷などを解析することによってロボットに必要な強度を決めるわけである。これは高速に走っていた自動車を急ブレーキで止めたときの解析にも当てはまる。

４．８　ＡＩ

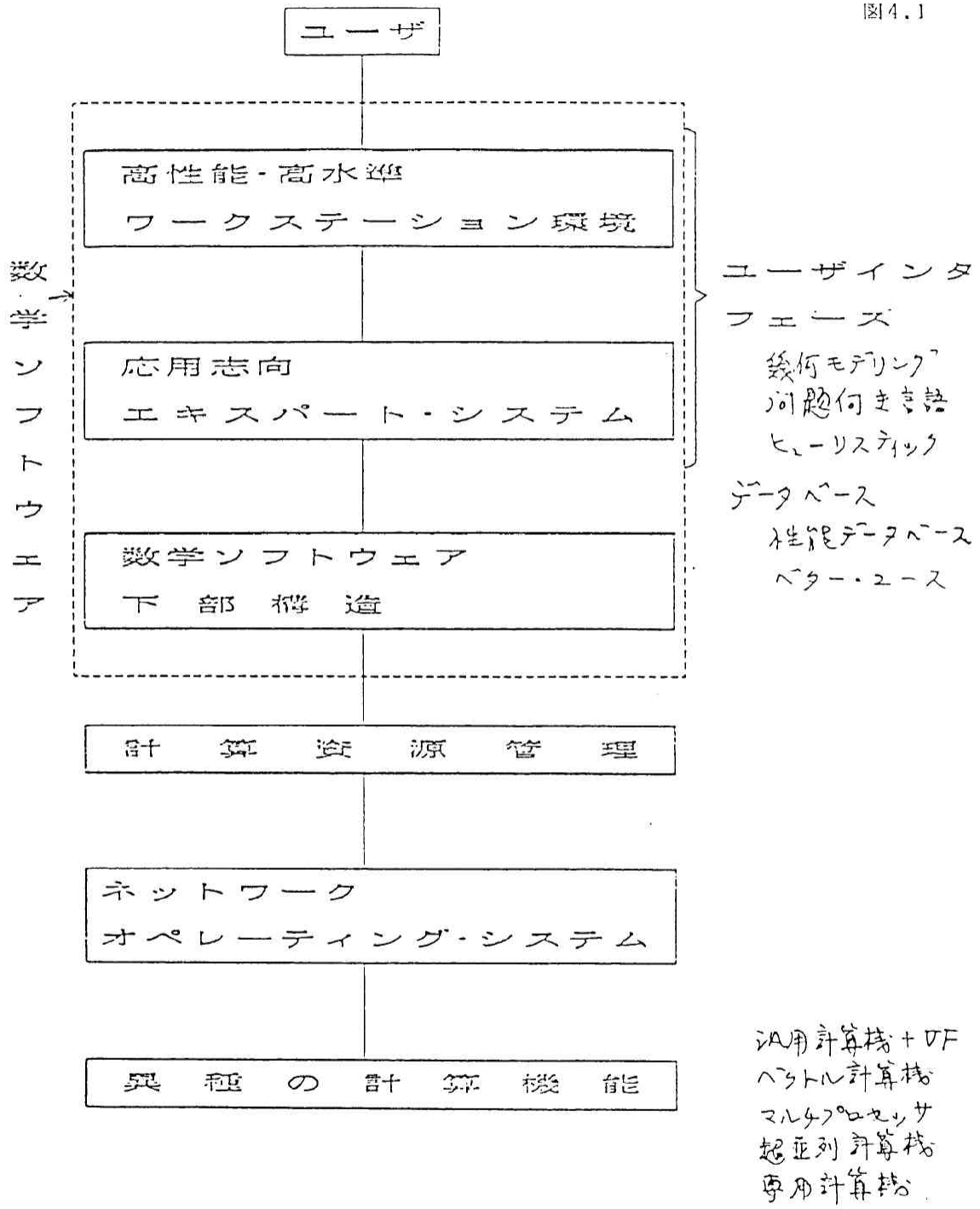
科学技術計算のすべての段階で、過去に蓄積されたノウ・ハウがある。これらのノウ・ハウは体系立ってはいないで、個々のユーザの頭の中や文献の中にとどまっている。これらの死蔵されたノウ・ハウを掘り起こし、一つの知識ベースとして既存の応用ソフトウェアに接続できれば、全体の処理効率を高めることができる。そのためには、高品質の技術系算用ルーチンの部品化がなされて、知識ベースの一部としてのルーチン・ベースに貯えられ、応用ソフトウェアがダミー・ルーチンの形で読めるようにしておく必要がある。応用ソフトウェアはすべての解法ルーチンを組み込むことはできないだろうから、せめて受け渡しの仕掛は用意すべきなのである。

また、モデルとアルゴリズムとハードウェアとの関係を示す実績データを性能データベースとして用意できれば、利用者は最も効果的な組み合わせをコンピュータ・ネットワーク上で選ぶことができるようになる。これらのデータベースや使用上のコンサルテーション技術は貴重なものであり、専門の技術者が必要となっていくであろう。

４．９　技術計算システム

いよいよ結論に移ろう。いままで述べたことを統合すると、図4.1と図4.2に示す、Purdue大学のRice教授の考え方に近づく。利用者とコンピュータをつなぐユーザ・インターフェイスとしては図形処理のできる高性能なワークステーション環境があると同時に、１章でのべた離散化と計算技法に関するノウ・ハウが知識

図4.1



科学計算の将来図（Rice教授）

ベースとして用意してあり、このためのエキスパート・システムが利用できる。最適化機能のように初期値のとり方によって特定の局所解に収束してしまう場合には、過去の経験情報をいれておくことも必要となろう。このような各種のデータベースの構築・維持管理には数値シミュレーションの専門家が必須となってくる。システム化がすすみ複雑になってくると、多分野の技術を見とおす専門家を育てておかないと、このような状況に対応できなくなる。いわば情報カウンセラが必要である。

さて、モデルは幾何形状を形成したり記述できる問題向き言語で現象を記述できなくてはいけない。それも単にプログラムが組めるというのではなく、対話的に利用者がモデルを作るのを援助してくれるかたちが望ましい。解析部分の数学ソフトウエア基本部はマトリックスのオーダリングや分割を行ったのち、ハードウェアの性能データベースを利用して、ネットワークにつながる最適とおもわれるコンピュータ資源を選んでデータを送る。場合によっては解析プログラムも送ることがあるが、普通は大型コンピュータ側にプログラムは用意されている。もちろん、ネットワークの利用にあたってはセキュリティや安全性が保たれていなければならない。

1980年代からソフトウェア・ハウスによる数値シミュレーション用ソフトウェアの生産が急速に高まった。60年代から70年代初期まではハードウェア・メーカやNASAが開発の中心であったが、いまや、大学を背後に抱えたソフトウェア・ハウスによる開発に中心が移りつつある。これらの数値シミュレーション・ソフトウェアの設計・製造・検証・保守・コンサルテーションについては、表4.1に示すような項目について検討しなければならない。コンピュータ・システムが作る側の論理が中心ではなくなり、使う顧客の立場からの観点が最優先になりつつある。したがって、標準化の推進や評価の実施、および顧客の現象把握から分析にいたるすべての段階でのコンサルテーション、が重要な意味を持つ。

最後に、数値シミュレーションの限界をよく認識しておいてほしい。数値シミュレーションの結果を無防備に使うと危険なのは、１章の原子炉の冷却シミュレーションでものべたとおりである。この点、図4.3に示した猪瀬教授の話はとても参考になる。筆者も最近の関心は科学技術計算システムを含めたソフトウェア・システムの安全性にある。安全性というのは、狭い意味では安定して動くという

図4.2

数値シミュレーション・ソフトウェアの
生産と販売

１．設計
（1）マルチベンダによる仕様の決定
→市場ニーズの吸収
（2）解析領域が広い→応野分野が広い
（3）数学的・物理的にしっかりしている

２．製造
（1）標準化のフォロー
（2）良質な数値計算ルーチンの使用
（3）入力データの会話機能
（4）出力データの標準端末への図形出力
→輸出
（5）データベースとの接続とデータの
格納
（6）シミュレーション機能
（7）エラーチェックの充実
（8）出力情報の言語からの独立 →輸出

３．検証とテスト結果
（1）理論値による検証
（2）他システムでの結果と比較
（3）性能面での評価（メモリ，速度，精度）
→ユーザの信頼

４．保守と機能向上
（1）フィールドテスト（ヒートランテスト）
→バグの枯化
（2）ユーザ協議会からの要求の吸収
→ユーザ志向

５．コンサルテーション
（1）マニュアルの市販本化→権威付け
（2）講習会の実施→ユーザ層の開拓
（3）コンサルティングスタッフによる指導
→モデル化と評価の指導

ことであるが、それだけではなく、システムを利用したり、その影響をこうむる人たちの安全性も含めて考えるべきである。数値シミュレーションにかぎらず、コンピュータの利用は方向として便利さを狙っており、自動カメラのようにバカチョン方式に移りつつある。しかし数値シミュレーションは、最終的には現象を近似的に解明するための道具にすぎないので、それを正しく使うには利用者の経験と見識によるところが大きい。結果が視覚化されてくると、近似計算にすぎないものが、100%現実のような迫力をもって迫ってくる。研究者や設計者の経験が全社的にある一つのデータベースに組み込まれて、利用できるとともに教育にも活用できることが望ましい。さらに、成功例というよりは失敗例を中心としたデータベースが構築できれば技術の伝承にもっとも効果が上がるとおもわれる。このためには、データベースの維持管理をして、その内容を絶えず最新の状態にしておくことと、内容の評価をすることが大切である。最近欧米では組織の内外の情報の取捨選択および連絡を担当する専門家（情報カウンセラとか情報ゲートキーパーという）が育ちつつあるという。科学技術計算システムに関してもこのような情報カウンセラが必要であり、世界の学者や専門家と公式・非公式なつながりをとおして情報を交換し、コンピュータのよき使い方について指導し発言できることが望まれる。

図4.3

情報技術の社会への影響（情報管理Vol126-1/1984）
猪瀬博東大教授

・技術的可能性大
→社会が必要とするかどうか技術屋に不明
→社会的要因（例．法制度の問題）
社会のニーズが左右
・技術のブラック・ボックス化
→技術がツールだという前提が忘れられる。
→ツールが悪いのか、使う人が悪いのか不明瞭になる。
・故障…絶対に壊れず、悪用もされないシステムはない。
→損害保険制度
→被害額、したがって保険料率の算定が大変難しい。
・文化面に対するインパクト
→文化の圧殺者と新しい文化の創造に寄与
・知的なコミュニティへの移行
（1）情報通信のネットワーク
（2）交通のネットワーク

結論
・技術はあくまでも人間のツール
→ツールで人の頭をたたいたり、自分の手をたたかないよう努力すること。
・経済性や便利性だけを追求していてもよいのか。