コンピュータシミュレーションのソフトウェア環境

佐久田博司*

Recent software environment in computer simulation

Hiroshi SAKUTA*

1. はじめに

シミュレーションに関わるソフトウェアの立場は、まったく異なる二つの方向がある。一つは、自力開発型というべきもので、もう一つは、ソフトウェアパッケージ利用型である。

前者は、研究者あるいはその支援者が基本的なシミュレーションの骨組み（構成方程式系、境界条件）や計算機資源（CPU 性能、メモリー容量）、開発環境（言語、入出力支援）などすべてを自分で判断して準備しようというもので、ある程度開発の経験をもったリーダーと経済的な基礎および開発期間を必要とすることは明らかである。逆に後者は、市販の汎用パッケージソフトウェアを購入してそのまま使用したり、支援システムを自作して、入出力や解析環境を改良したりすることによって目的に沿って利用するものである。

シミュレーションの利用を試みる場合に、最初に突きたる判断分岐がこれらのどちらの道を選択するかとい

うことである。もちろん、両者の中間に相当するものは、数式処理言語を利用したり、統計処理専用言語の上で自分のシステムをつくるといったことで存在するが、本解説においては、「自作」と「利用」という両極端を、主に例として説明する。

また、昨今のパッケージ型ソフトウェアは、機能が高度化しているために、そのモデルデータ作成など、「利用に関係する作業」自体がプログラム開発の側面を有していることも事実である。

この条件分岐をどちらに選択するかで、以下の議論はまったく異なることになるので、その判断の基準を表 1 に表す。

これをみると、大まかに言って、短期的な定型業務にはパッケージ型、長期的に技術力を蓄積する目的には開発型と分類することができる。シミュレーションプログラムの開発は、技術的なトレーニングとしては効果が大いといわれている。そこで、本格的な解析の前に、単純な系や現象について小さなシステムを自作すること

表 1 自作/パッケージソフトウェア比較

	自作システム	パッケージ型ソフト
初期コスト	低い	高い
立上げ(利用開始までの期間)	比較的長い	短い
保 守	開発者に依存	メーカーに依存
長 所	内容について十分理解することができる。 開発を通じて研究者の技術レベルが向上する。 販売できるかもしれない	他で実績があるもの、競争原理によって淘汰されたものを選ぶことができる。 メーカーのサポートを受けられる。 他の研究/設計機関との Cross check 可能。
欠 点	Bug は常に自分の責任で処理する。 開発コスト高い。	大規模なものにたいしては処理コスト高い。 技術レベルの向上は多くは期待できない。

* 青山学院大学理工学部機械工学科（東京都）。Dept. of Mechanical Engineering, College of Science and Engineering, Aoyama Gakuin University (Setagaya-ku, Tokyo).

受理日：平成 5 年 8 月 4 日

によって、パッケージ型ソフトウェアの選定の予備知識を得るなど、準備に相当する段階を考えることもできる。それぞれについて、利用するときの注意を以下にまとめる。

パッケージソフト利用の原則

- (1) 操作方法を覚えるだけではなく、問題の分析能力を向上させる手段としてとらえる。(Operator を養成するのではなく、Analyst を育てるつもりで)
- (2) 互換ソフトウェアの動向を常に視野に入れる。
- (3) 出力は提供されたものだけではなく、独自の判断が必要である。
- (4) 鵜呑みにしない。
- (5) Cross check を常に怠らない。できれば、複数のソフトウェアで同じ問題を解析してみる。

自作システム開発の原則

- (1) 総合的な解析能力の向上を目標とする。最初は、短期的な問題解決の手段としては考えない。
- (2) 大規模な開発では適切なグループリーダーが成否を左右する。
- (3) 完成までの目標期間の設定が重要である。
- (4) 実験などとの比較検証手順の計画を事前に立案する。
- (5) 入出力機器や、支援システムなど周辺の知識は重要である。

シミュレーションのためのソフトウェアは、一般に数値解析を行う本体部分（ソルバー）と、入力データを解析対象の条件に従って作成したり出力を可視化したりする周辺部分（モデラー）で構成されている。ソルバーは数値演算操作そのものだから、技術的な進歩はほぼ計算機の CPU 基本性能で決定してしまうが、モデラーについては、その構成方法や周辺技術に関する知識がそのまま性能に反映する。一般に、ソフトウェア技術は、ハードウェア環境の進展に比べて、まだ立ち後れている段階にあると思われる。それは、ソフトウェアが工学の分野として影響を他の分野に及ぼし始めたのが最近であり、まだソフトウェアエンジニアの教育が本格的に行われていないためであろう。しかし、ソフトウェア技術の進歩が、モデラーなどシミュレーションの周辺技術に及ぼす効果は大きいと見込まれるため、今後、パッケージ型のソフトウェア供給が広まっていき、それに従って、シミュレーションの果たす役割は大きくなっていくと判断できる。

本稿では、まず、パッケージ型ソフトウェアと開発型ソフトウェアの概要を比較し、次に両者に共通する技術の現状を記すこととする。

2. パッケージ型ソフトウェア

2.1 選定の基準

数値解析の能力も、解析手法に関する知識の蓄積と、計算機の性能の発展が、汎用のパッケージ型ソフトウェアに結実しているといえることができる。1960年代から70年代にかけて急速に開発の進んだ解析アルゴリズムは、現在、ほぼ完成の域に達しており、最近の技術トピックスは、ほとんど解析対象の新しさに求められている^{1),2)}。たとえば、FEA (Finite Element Analysis) の中でも線形解析はごく日常的な設計作業の中に組込まれており、プラントの安全審査などでも標準化されたツールとして使われている。しかし、材料の力学的特性や温度依存性のある物理定数など、非線形の特性を考慮した、いわゆる非線形解析では、まだ、いろいろな分野で適用性を試されているところである。したがって、特殊な分野を除いて設計業務などで FEA を必要とする場合、市販のパッケージソフトウェアの内容を吟味して適用性について検討するという作業は、最初に行うべきことの一つであろう。

はじめに述べたように、自作ソフトウェアによってこれらに対応しようとするのは、コストと時間の点で不利であることは確かである。

とはいえ、どうしても自作の必要がある場合、簡単なモデル、たとえば、2次元解析で数1000要素（自由度）程度なら自力開発も不可能ではないが³⁾、多少の汎用性（他のモデルにも適用できる）を追求したり、実用的な3次元解析を行おうとするならば、自力で開発することは、いまや研究の範囲においてすら現実的なものではなくなっている。自作しようとする場合でも、まったく新しいアルゴリズムを組立てることはほとんどないため、汎用コードの開発の跡をなぞるような作業となるわけである。

更には、開発のための人材、時間については、開発したプログラムの実用性に対応して相当のものがなければならない。現在では、計算機資源の面で比較的余裕ができたため、特に開発のための「人材」の確保が、ほぼ成否を決定するといっても過言ではないだろう。「人材」といっても、数値解析を専門とする大学の研究室は比較的限られており、供給量の大変に少ない資源であるといえる。

汎用コードで実際に利用されているものは、FEA が圧倒的に多く、その適用分野も様々である。しかし、最近流体力学特性や放熱特性など高度な解析を要求する設計が増えてきたため、数値流体力学と呼ばれる学問分野も認

2

表2 大型計算センターに揃えてあるプログラム例
(東京大学大型計算機センターの場合)

数値計算	LAPACK	行列計算ライブラリ
	SPARSPAK	疎行列直接解法ライブラリ
	ELLPACK	楕円型偏微分方程式解法用ライブラリ
熱・流体解析	PHOENICS	汎用熱流体解析コードシステム
	α -FLOW	汎用3次元流体解析システム
	FLIDAP	有限要素法非圧縮性流体解析プログラム
	FLUENT	汎用熱流体解析プログラム
構造解析	MARC	MARC Analysis Research Co. USA
	ADINA	ADINA ENGINEERING Inc. USA
	ABAQUS	Hibbitt, Karlsson & Sorensen Inc. USA
	BEASY	CM Beary Ltd. USA
	NASTRAN	MSC/NASTRAN USA

知されるようになってきており、差分法その他の手法もそこで活用されている(表2)^{4)~6)}。以下に、それぞれについて、現在流通している汎用コードの特徴をまとめて記す。

2.2 有限要素解析コード

FEAは、利用の歴史も長く、事例が多いことから信頼性も高く、様々な形で商品として販売されている。表3は、これらのうち代表的なものに含まれている解析手法を、材料製造方法によって分類したものである。明らかに、比較的簡単な微小変形範囲の応力-ひずみなどの線形解析のような機能については、ほとんどコードによる差はなく、価格などで選定することができる。

操作性やデータの整合性チェックなど、本来の解析アルゴリズム以外の機能が選定の基準となることが多くなってきた。また、最近では、材料加工など大変形や接触問題と呼ばれる固体間の相互作用、材料と境界条件の非線形特性を扱うことが可能となったコードも増えている。

汎用 FEA コード MARC によるフランジの加工解析の例を図1に示す。入力データは一般に解析対象が複雑になるにつれてその量が増えるのが普通で、いわゆる「手入力」によって、座標データや要素間の結合データなどを作成すると、誤入力の訂正、すなわちデバックに相当の時間を要することになる。

そのため、コードによっては、専用の入力プログラムや、メッシュ生成プログラムを用意しているものもある。

表3 金属材料に関する解析手法の比較

加工方法		弾塑性解析	塑性流れ	熱・応力解析	相変化	摩擦	接触問題
鋳造	凝固			○	○		○
	鋳型湯流れ			○	○	○	
塑性加工	圧延	○	○	○			○
	鍛造	○		○		○	
	押出し 絞り	○		○

図1 汎用有限要素法コードによる加工解析例

る。最近では、この入力作業を解析すべき対象の物理モデル化という意味で一般性の高い汎用入力ツールの研究も行われている。FEAは、そのモデルデータの作成が解析作業のほとんどであるため、この作業の知識化が進められており、今後、モデルのデータベース化などの課題が残されている^{7),8)}。

このほかに、供給するソフトメーカのコンサルティング体制なども考慮すべき問題であろう。プログラムのクロスチェックや、参考になる例題の探索、情報交換を目的として、ユーザーズがネットワーク上でグループ化する方向も現れてきているが、これに対するメーカ支援体制も評価の一つになるだろう。

図2 開発プログラムによる流れ解析の格子生成例
(a)変形したJoukowski翼, (b)円柱まわり,
(c)広がり管内

2.3 流れ解析コード

流れの解析, ナビエストークス式などの運動方程式を数値的に解くというアイデアは今世紀初頭から提案されていたものの, 実際の現象と比較できる実用性の高いものとして発表されたのは, Los Alamos 研究所のグループによるものが最初といえるだろう⁹⁾。速度場や圧力場, 温度場などの連続体に対して差分と呼ばれる離散化を行い, 運動方程式などの偏微分方程式を差分方程式に書き直し, 離散化量の連立方程式をたてて, それを解くことによって各場の解を得ようというもので, 基本的な骨格は Los Alamos のグループが提案した方法が現在まで使われている^{10), 11)}。差分化の手順の中に, 離散化を矩形の格子で行う必要があるため, 境界の形状がなめらかなものとするのが難しかったのが大きな難点だったが, 方程式の離散化の途中に, 数値的な写像を取入れることによってこれを解決できるようにしたのが, 流れの差分法による解析を一般的にした理由と思われる。図2にその格子の例を示す。いずれの格子系も写像空間においては長方形格子となる。コード名としては表2に挙げたものが有名である。

図3に解析上の最も注意すべき問題であるモデルの複雑さ(分割数)と解析結果の例を示す。明らかに分割の少ないモデルでは現象が正しくとらえられていないこと

図3 開発プログラムによる弾性振動板回りの流れの解析例
 ξ : 円周方向分割数 η : 半径方向分割数

がわかる。先に述べた Los Alamos のグループは, はじめてコンピュータシミュレーションを, アニメーションとして可視化する試みを行い, 大きな成功を収めたこともよく知られている。

3. 開発型ソフトウェア

3.1 プログラミング言語

次に, プログラムを自作する立場からいくつかの乗り越えなければならない障害について説明する。

自作プログラムを作成しようとするとき, 最初に思い浮かべるのは, 高級言語(人間が解読しやすいソーステキストの形でプログラムが記述できる言語体系)で書かれたソースプログラムである。1980年代までは, シミュレーション用の高級言語といえば, FORTRAN を使うのがごく普通であった。FORTRAN は, 数値計算向きのプログラムの最適化機能が標準的に備わっていて, システムまわりのことはほとんど考えなくて良いということから, 当初, ほかに高級言語を使うことはあまりなかったようである。ところが, シミュレーション用計算機資源が, ワークステーションを中心とした Unix システムに移行するにつれて, C や, Pascal などの ALGOL 系の言語を使うようになってきた。以下にこの高級言語と特徴をまとめる。

3.1.1 FORTRAN

1980年代までは, 数値シミュレーションと FORTRAN 言語はごく自然に結びついており, コンピュータ言語の専門家でなくても簡単にプログラムを組むことができることから, 研究者などの間では標準言語としての地位は揺るがなかったといえる。なかでも, 細かいシ

2

システム制御やファイル入出力、グラフィック処理が要求されなかったことが、FORTRAN が長く利用された理由と思われる。

その言語としての発展を生物進化的にみると、まず、汎用機の時代にラインごとに1命令という、バッチ処理+カード型ソースの時代の最適な言語として発生し、解析者の計算機利用環境が広がるにつれ、Unix ワークステーションや、パーソナルコンピュータに移植され、完成の域に達した。何回かの改訂を受け、ANSI で、FORTRAN77として1977年に仕様が決まったあとは、ほとんどそのままの形で機能を継承していった。

現在、完成度は十分高い段階にあり、高速化のためのコンパイラの最適化処理は、ほぼ完全になっている。パソコン環境でも、十分に実用的なコンパイラが供給されており、高速のPCはシミュレーション用資源としてプログラミング言語の点でも十分実用に耐えるものとなった。

3.1.2 C

現在、プログラミング言語としてもっともよく使われているのはCだが、つい最近までは、シミュレーション用の言語の候補としては注意が払われていなかった。これは、Cがシステムプログラム用の言語であるという先入観が強かったためもあるが、主に、研究者がそれまでにFORTRANで開発したプログラムの資産が役に立たなくなるというのが理由であった。確かに、C言語自身が持っている数学関数などは数値計算向きであるとはいえないのだが、プログラマおよびプログラミング教育が全体にCに移行していることと、関数ライブラリなどの整備が進んでいること、スーパーコンピュータなどでもCがサポートされ、環境が整ってきたことから状況は徐々に変わっている。しかし、古典的な関数プログラムは、FORTRANライブラリの形で保存されていることが多く、当然両者は並存していくと思われる。

3.2 並列化処理

スーパーコンピュータは、数値計算において現れる高精度実数計算を何らかの形で高速化しており、通常は、計算手順の並列化によってそれを図っている。しかし、高速化率を向上させるためには、プログラミングにおいて多少の技術が必要で、並列化率を上げなければならないために、完全に最適なプログラムをかくことはたいへんに難しいとされている。並列処理専用のプリプロセッサなどが考えられるが、有効なものはまだないようである¹²⁾。

3.3 ライブラリ

FORTRANは、数値計算を主な目的とした言語であ

図4 ソフトウェアの移植性とユーザインターフェース

るため、数式に記述される記法と似た形で関数ライブラリやサブルーチンを使うことができる。三角関数などの単純な数学関数以外にも、大規模な連立方程式の数値解を求めるサブルーチンや補間法、最適化なども用意されている。更に、数値解析については常微分方程式のためのパッケージソフトウェアなどが標準的に供給されている(表2参照)。これらをうまく組み合わせることによってプログラム開発に要する労力は大幅に削減されるはずである。Cについてもほぼ同様なことがいえるようになってきた。つまり、Cから直接FORTRANのライブラリをコールしたり、ライブラリを移植(Cで書き直す)することによって、FORTRANによる開発環境に近いものが得られるようになってきた。ただし、高度な機能をもったサブルーチンやライブラリソフトウェアは、汎用機やスーパーコンピュータでは備え付けになっているが、ワークステーションやパソコンでは必ずしもそうではなく、購入したり、移植したりすることもある^{13),14)}。

その他、グラフィックス関連の関数は、計算機の機種に依存するところが大きいので、特にライブラリの役割は重要である。ライブラリが高度に発展していくと次節に示すような専用の言語としてまとめられていくことになる。

3.4 移植性

プログラムはどのような目的であれ開発を進めていく過程で必ず通る段階がある。それは、「移植(Convert)」と「更新(Revise)」と呼ばれている。「移植」とは、異なった機種の計算機である場合もあるし、他の言語への場合もある。

図4はユーザが扱う最も上位のプログラムソースを、機種や出力機器になるべく依存しない形に記述することによって移植性を高める方法を示している。

4. シミュレーション用周辺ユーティリティ

4.1 汎用コード用入力データ

汎用コードの入力は様々な条件を入力するという一般性を考慮して、カード型のデータをキーワードでブロック化したものが用いられていることが多いようである。これは、大型汎用機に入力する場合のカードのスタイルが保存されているという歴史的な経緯と、編集のしやすさからであると思われる。これを解読するのは、熟練者にもかなりの負担を強いる困難な作業である。作成したモデルから入力データを構成することが、計算機のハードウェアの進展に伴って、大きな比重を占める作業となってきた。更に、FEA では要素の座標値や結合状態を示すためのデータなど大量の数値データを必要とするため、専用の処理プログラム（たとえば MENTAT : MARC 社）が使われている。

4.2 プリ/ポスト プロセッサ

入力データの作成と解析結果の表示について専用のプログラムによって統一的に扱うという目的で、各コードはプリ/ポスト プロセッサを提供している。最近では、表示用グラフィックツールがつくりやすい Unix ワークステーションの上で、X 環境を基礎とすることが多いようである。また、大型の計算を行う場合は、プリ/ポスト プロセッサは、必ずしもソルバーと同じ計算機の上にあることがなくなり、ネットワーク上でソルバーの CPU 性能を阻害しないように別の表示専用マシンを使うのが最近では普通の構成となっている。

5. グラフィックスの基本

現在、シミュレーションとその表示は、「流れの可視化」などからの連想で、「シミュレーションの可視化」という場合もあるほどで、正に数値実験の領域にシミュレーションが到達していることを示している¹⁵⁾。

グラフィックスの機能を充分利用するためには、従来、機種ごとに異なった関数など利用する必要があったが、GKS などを経て、現在主流となっている Unix-X Window 環境において、X-lib と呼ばれる関数群が統一的な利用を助けている。その他、パソコンでは言語に付属した関数がこの役割を果たしている。しかし、これらをそのまま利用してグラフィックス機能をもつプログラムを開発することも、初心者にはかなりの負担なので、専用のパッケージソフトウェアが販売されていたり、PDS*として公開されている。

5.1 X 環境

統一的なグラフィックス環境は、WS では、X 環境が

最も基本的なものである。これによって、Unix 上の共通資産であるプログラムの互換性が向上し、さらにコードの保守が楽になったと言われている。現在、Unix のグラフィックス環境は、X あるいはその系統 (Motif, Open window etc.) ではほぼ統一されている。

5.2 その他の表示ツール

(1) アニメーション

動解析を行った場合時間方向に結果を見るためには、アニメーションのような方法が、人間の認識能力に最も適している。ただし、現在でもシミュレーション結果を手軽にアニメーション化してビデオテープなどに変換するシステムは一般的でないため、コンソール上で疑似的に動画化するのが普通である。

(2) ページプリンタ・プロッタ

ページプリンタは、PDL (Page Description Language) と呼ばれる専用の言語を持っている場合、高度な出力を得ることができる。現在世界で流通している主な PDL は、① Postscript, ② LIPS, ③ ESC/PAGE などである。

6. 基本ソフトウェアツールの入手方法

前章までに述べてきたフリーソフトウェアなどは、自由に入手することができるが、その入手経路はそれぞれ異なる。以下に、シミュレーションの Post 処理に関係する主な PDS ソフトウェアとその入手方法について述べる。

(1) NGRAPH PC 用 2 次元グラフユーティリティは PC-9801 (NEC) シリーズで利用可能で、最小 2 乗

* PDS (Public Domain Software) : 公開ソフトウェアとして著作権を放棄したもの。Unix で最も広く使われているエディタである “emacs” や、シェルスクリプト言語として最近多く使われている “Perl”, 清書ソフトウェアとして有名な “Tex” などが代表的なものである。“Tex” の著者 Knuth らが、積極的に著作権を放棄することでソフトウェア供給の流れを大きく変えたといわれている。同じように公開するが、著者による一定の保守と実費を考慮した “Shareware” や、ネットワークから Down load するときに実費請求する場合がある。“Freeware” は著作権を留保した公開ソフトウェアである。ソフトウェアの著作権は、国内外でその保護方法が議論されており、まだ明確な決着を見ていない。PDS の形で個人の著作権を放棄することによって、世界のソフトウェア環境を改善した功績は大きいと評価されている。

反面、出所が明らかでないソフトウェアなどによってコンピュータウィルスがまき散らされるなどマイナス面も目立つようになってきた。

図 5 NGRAPH によるグラフ出力例

近似などデータの加工についても標準的なものは揃っている（作者：石坂 智）。特に、ハードコピー出力は、清書用の高品位のプリンタ（Postscript, LIPS）が使えることが特徴である。図 5 がその出力例である。入手は [NIFTY SERVE FSCI #43] などパソコンネットワークから可能である。

(2) GNUPLOT GNU (グニュー) プロジェクトは、R. Stallman によって提唱された FSF (Free Software Foundation) が提供している一連のソフトウェア群で、GNU-Emacs (Unix 上で最も多く利用されているエディタ) や、GNU GCC (GNU 版 C++) はか数多く利用されている。GNUPLOT は、その中でデータの 2/3 次元グラフ表示などを行うことのできるツールで、PC にも移植されている。入手については、表 4 の FTP サイトから転送することができるが、直接表 5 の FSF から入手することもできる。

(3) tgif+ Unix 上で動く簡易 CAD ツールで、構成図やレポートを書く場合に有効である（作者：W

表 4 Anonymous FTP が可能な計算機

地域	アドレス	機関
関東	utsun. s. u-tokyo. ac. jp	東京大学
京都	ftp. kuis. kyoto-u. ac. jp	京都大学
大阪	ftp. ics. osaka-u. ac. jp	大阪大学
九州	ftp. csce. kyusyu-u. ac. jp	九州大学

Chiaw Cheng, 日本語化 + α : 松田晃一)。図 4 もこの tgif+ を使って書いたものである。

7. おわりに

ここまで、自力開発とパッケージ利用というまったく異なる方向からシミュレーションに関わるソフトウェア環境を見てきた。しかし、実際の場面では両者を区別するよりもそれぞれの利点をうまく使い分けることが重要である。

汎用パッケージも大規模のコードのままの供給だったものが改善され、より細かなカスタマイズ（経済的なコードサイズや最適 CPU の選択など）が達成されるだろうし、ユーザカスタマイズ（ユーザの希望に沿ったコードの再構築）も大きく進歩すると思われる。開発型のプログラムについても、すでに開発されたプログラムを組合わせていく作業が中心の時代に入っていく傾向にある。

EWS の性能向上やスーパーコンピュータの普及により、シミュレーションの重要性は増していくだろう。そのためにも、現在のソフトウェア資源の状況を理解し、より一層有効に活用することが大切になると思われる。

本稿の構成について、多大なご助言をいただいた東京大学工学部鈴木俊夫助教授に心から感謝する。

参 考 文 献

- 1) 鷺津久一郎ほか：有限要素法ハンドブック，培風館，(1981)。
- 2) ツィエンキービッツ（吉識雅夫監訳）：マトリックス

表 5 シミュレーションに特に関係の深い学会および団体

学 会 名	連 絡 先	電 話
日本シミュレーション学会	〒 151 東京都渋谷区千駄ヶ谷5-10-11 (財)日本科学技術連盟内	03(5379)1236
(株)可視化情報学会	〒 169 東京都新宿区百人町2-16-13	03(3364)1762
Free Software Foundation	675 Mass Ave, Cambridge, MA 02139, USA	+1-617-876-3296

- ス有限要素法, 培風館.
- 3) 三好俊郎, 白鳥正樹: 演習有限要素法, サイエンス社.
- 4) 機械学会編: 熱と流れのコンピュータアナリシス, コロナ社, (1988).
- 5) 機械学会編: 固体力学におけるコンピュータアナリシス, コロナ社, (1988).
- 6) 機械学会編: 振動工学におけるコンピュータアナリシス, コロナ社, (1988).
- 7) 矢川元基編: 計算力学 [I] -新しい応用と展開-, 養賢堂, (1989).
- 8) 矢川元基, 福田収一編: 計算力学 [II] -計算力学とAI-, 養賢堂, (1989).
- 9) ローチェ: コンピュータによる流体力学 上/下/演習, (高橋亮一編), 構造力学研究所, (1981).
- 10) 保原 充, 大宮司明: 数値流体力学-基礎と応用-, 東京大学出版会, (1992).
- 11) 機械学会編: 流れのシミュレーション, コロナ社, (1988).
- 12) 長島重夫, 田中義一: スーパーコンピュータ, (電子情報通信学会編), オーム社, (1992).
- 13) 森口繁一ほか: 電子計算機のための数値計算法, 培風館, (1965).
- 14) マコーミク (清水留三郎訳): 数値計算プログラム, サイエンス社.
- 15) 日本図学会編: CG ハンドブック. 森北出版, (1989).