Vol. 28 No. 10　　情　報　処　理　　Oct. 1987

解　説

3. 応用分野の最前線

3.2 技術計算分野における自動プログラミング†

梅谷征雄††　金野千里††
山本富士男††

1. まえがき

技術計算を自然現象の解明ならびに工学面の応用のための計算と位置付けるとその種類は多岐にわたるが，ここでは特に数値シミュレーション[1]を取り上げて解説することにする．数値シミュレーションとは“物理現象”の“数値シミュレーション”の意であり，偏微分方程式などによりその挙動がモデル化される物理現象をスーパコンピュータなどの大型計算機内に再現することにより，実験では不可能な環境設定を行ったり観測不能な詳細情報を得ようとする技法であり，科学技術の巨大化，計算機の高性能化とともに，最近とみにその重要性を増しつつある技術である．またこの分野の計算は長い歴史をもち，比較的手法の標準化が進んでいることから，さまざまなレベルでプログラム作成労力を軽減するための自動プログラミングの試みがなされている．本解説では，数値シミュレーションをめぐる自動プログラミングの諸技法を概括した後，厳密には自動プログラミングの範疇には含まれないが従来からの伝統的な省力化技法であるマトリクスライブラリと最近特に進展の著しい偏微分ソルバを中心に事例を紹介することとする．さらに最近の話題である“統合問題解決支援環境”(Problem Solving Environment) と呼ばれるこれら諸技法の統合の動きにふれることにする．なお，自動プログラミングの重要な項目に数式処理があげられるが，これについてはまとまった解説[2]があるのでそちらを参照願いたい．

2. 各種自動プログラミング技術とその位置付け

2.1 数値シミュレーションシステムの構成

地球表面の大気の流れや電子管内に形成される電位分布，あるいは半導体デバイス内部の不純物分布などを計算の対象とする数値シミュレーションシステムの標準的な構成は図-1 に示すようになる．すなわち，これらのシステムは一般に，シミュレーションをとり行うべき空間領域の形状などを指定する入力処理部（プリプロセス）と中心となる解析処理部およびシミュレーションの結果を図形や画像に編集して分かりやすく表示する出力処理部（ポストプロセス）から構成される．

図-2 に典型的な数値シミュレーションの例題として電子管内の電位分布の計算例を示す．管は軸対称とし，軸を含む断面の分布を示している．ここで電位分布はラプラス方程式 $\Delta\phi=0$ に従うものとし，管壁における適当な境界条件のもとで解き内部の電位分布を決定している．入力処理部は管壁の形状などを図形端末より入力し計算処理の可能な内部表現に変換する．解析処理部の出力である電位分布は空間代表点の離散的な数値で得られるので，それをもとに図-2 のような等高線図を描き連続的な描像を復元するのが出力処理部の役割である．

ここで，入力処理部と出力処理部は機能が固定的であることから，でき合いの機能ソフトウェアが利用されることが多い．これに対して，解析処理部は対象ごとに個別に開発しなければならないので，最も工数が掛かり自動プログラミングへの要望が強い．この部分に対する自動プログラミングの手法を次に考察する．

† Automatic Progromming in Scientific Computing by Yukio UMETANI, Chisato KON'NO and Fujio YAMAMOTO (Hitachi Central Research Laboratory).
†† (株)日立製作所中央研究所

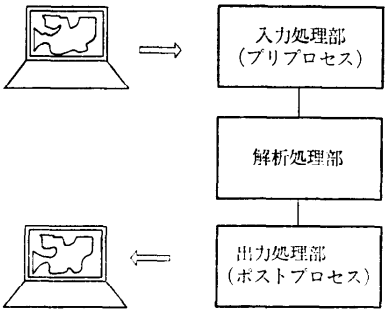


図-1 数値シミュレーションシステムの構成

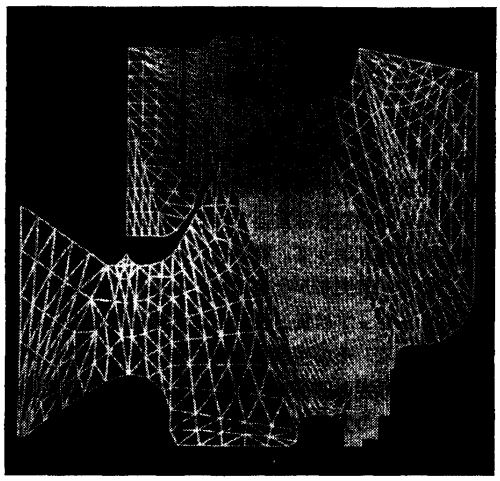


図-2 数値シミュレーション例（電極内電位分布計算）

2.2 解析処理における自動プログラミングの手法

解析処理部のプログラム作成手順とそこでの自動プログラミングの手法，ならびに自動化の範囲を図-3に示す．

プログラム作成手順は大きく，定式化，スキーム化，プログラム化の3段階に分けられる．定式化段階では物理現象を偏微分方程式などを用いてモデル化する．図-2の例では電位の分布がラプラス方程式と境界条件に従うものとしてモデル化を行っている．続くスキーム化の段階では，微分方程式を差分法，有限要素法，境界要素法などの手法を用いて空間内の代表点（離散点）の間の関係式に置き換える．この操作を離散化と呼んでいる．計算機の有限の資源を用いる以上，離散化は避けられない操作である．図-2の例に対してガレルキン式の有限要素法を用いて離散化を行うと φi を基底関数ならびに形状関数とするとき，ラプラス方程式は以下のような離散点間の連立一次方程式に置き換えられる．

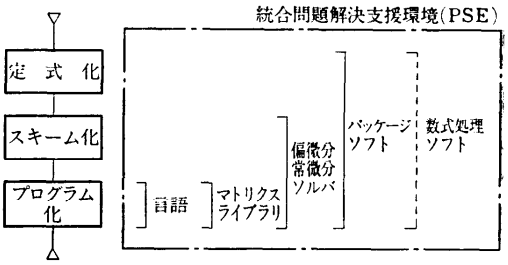


図-3 解析処理部における自動プログラミングの諸手段と範囲

$$Mz=b$$

$$Mij=\int_{\Omega}(\nabla\varphi_j\cdot\nabla\varphi_i)\,dx$$

$$bi=\begin{cases}0 & (i \text{ が内部離散点})\\ gi & (i \text{ が固定境界離散点})\end{cases}$$

このように数値シミュレーションの解析処理は大規模な連立一次方程式の求解に帰着されるケースが多い．未知量に関して非線型な方程式の場合にはなんらかの線型化近似をほどこして反復・収束計算を行うことが多い．このような計算手順の組み立ても離散点の設定と並んでスキーム化段階の重要な仕事である．最後のプログラム化段階では，スキーム化の結果を受けて連立一次方程式の係数行列と右辺のベクトルを計算し，引き続いて連立一次方程式を解くための計算プログラムを作成する．

このような解析処理の諸段階に対して，最も早く自動化（省力化）が着手されたのはプログラム化のフェーズである．高水準言語FORTRANを用いることによりアセンブラ言語に対してプログラム作成効率が大幅に向上し，また機種間の可搬性（portability）が高められた．一方，連立一次方程式の求解部分は特に汎用性が高いので，マトリクスライブラリ[3)~5)]として標準的な機能ソフトウェアが準備され，プログラマが逐一作成しないでも済むようになり大幅な省力化が達成された．

スキーム化段階の省力化の歩みはそれに比べて遅れていたがようやく最近，離散化過程の自動化を達成するものとして偏微分ソルバなど[6),7)]が出現しつつある．一方，定式化段階も含めて解析過程全体の省力化をはかる仕掛としては適用分野ごとのパッケージソフトがあり，これは古い歴史をもっている．さらに，ある段階の手順全体を自動化するものではないが，全段階にわたって局所的に有用なツールとして各種の数式処理システム[2)]があり，広く用いられている．

2.3 統合問題解決支援環境 (Problem Solving Environment)

特に最近の新しい動向として，上記のソフトウェアの機能を包括した上で，マンマシン性の向上とガイダンス機能の充実をはかり，ワークステーション上で働く数理エキスパートシステムを構築しようとするものがある．これは統合問題解決支援環境 (PSE : Problem Solving Environment) と呼ばれている[8)~10)]．PSE の目的は，これまでに述べた諸技法を自在に組み合わせて適用し，また対話的にその効果を検証できる環境を用意することにより，問題解決に至る総工数を低減することにある．

ここでは一昨年の(仏) Sophia-Antipolis における PSE に関する学会の基調講演[8)]からその概要を紹介する．

図-4 は本学会で議長を勤めた英国 Numerical Algorithm Group の B. Ford 氏が提案する PSE を示す．本システムは4つのセクションから成る．最左端のセクションは人間とのインタフェースの部分であり，テキストのほか，画像・音声などによる入出力機能を含む．左から二番目のセクションは問題解決のためのコンサルティングセクションであり，課題ベース，知識ベースとそれにアクセスするための言語システム (TLS)，コンサルテーションシステム (NCP) を含む．それに引き続く右から二番目のセクションが機種に合わせた実際のプログラムを生成する部分であり，前節 **2.2** のソルバ以下のソフトウェアに対応する．標準化されたプログラムモジュールのほかに，ユーザプログラムや分野ごとの計算スキームの原型 (Program Template)，機種に応じた変換規則 (Transformation) などのデータベースを含む．右端は実際に計算を実施するセクションである．

3. 事 例 報 告

本章では，前章で概括した自動プログラミングの諸手段から，プログラム化部分を自動化するマトリクスライブラリ，スキーム化を含めて自動化する偏微分ソルバ，および統合問題解決支援環境を事例により紹介する．プログラミング言語については特に FORTRAN 8X が重要であるが他に詳しい紹介がある[17)]のでそちらを参照願いたい．また定式化を含めたパッケージソフトは内容が多岐にわたるので省略させていただく．

3.1 マトリクスライブラリ

技術計算のプログラミングにおいて，数値計算ライブラリのもつ役割りは重要である．最近の高速計算機の急速な発達によって，科学技術の広範な分野にわたり，技術者・研究者が生産に携わる必要のある多様な数学ソフトウェアは質，量ともに大幅に過去のそれを上回っている．その生産性の向上と，作られるソフトウェアの信頼性の確保は，部品として使われる数値計算ライブラリの質に大きく依存すると考えられる．

数値計算ライブラリとしては，FORTRAN 言語処理系などに備えられている標準関数，特殊関数のほか，線形計算，数値積分，常微分方程式の求解，代数方程式の求解，フーリエ解析などがある．これらは各計

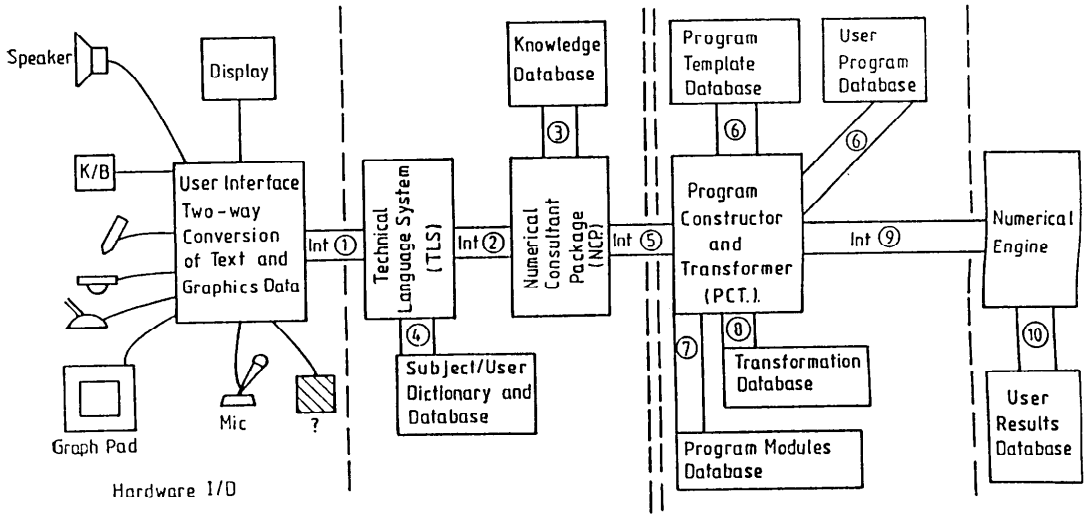


図-4 PSE (Problem Solving Environment) の一構成例[8)]

算機メーカ，大学などの研究所から多数提供されている[3),4)]．これらのライブラリが満たすべき要件のうち，最も重要なものはやはり高い精度である．さらに，計算規模の著しい増大にともなって計算の高速性に対する要求も高いものになっている．また，ユーザの日常的使用に十分耐えるためには使いやすさが重要であり，一方，自分個有の問題に特に効果的に用いようとする高度なユーザにとっては，ライブラリの部分的変更，計算結果の妥当性確認を可能とするため，計算アルゴリズムの明解性などが必要である．またライブラリを管理する立場からは，部品の品質維持，機械独立性の保持，種々の精度用のプログラム自動生成などが重要な課題である．

このような要件を満たす質の高い数値計算ライブラリの開発には高度な専門家を含めた多大な労力を要することが知られている．たとえば，1970年代前半に米国 Argonne 国立研究所を中心に開発された線形計算ライブラリのうち，固有値計算ライブラリ EISPACK の開発には，その第1版（34ルーチン，約 6,000行）の場合，博士クラス 96 人月を含む 126 人月を要し，さらに第2版（70ルーチン，約12000行）には，109 人月を要したという[13)]．

ここで紹介するマトリクスライブラリは，特に線型の偏微分方程式や常微分方程式の離散化の結果得られる連立一次方程式や固有値問題を解くための数値計算ライブラリであり，係数値を行列形式で保持することからこの名がある．

マトリクスライブラリには非常に多くの種類があるが，ここでは，上記 EISPACK と同様，1970年代中期に Argonne 国立研究所で開発された連立一次方程式用ライブラリ LINPACK[5)] について紹介する．

（1） LINPACK の目的と方針

数値計算ライブラリ生産手段の研究が第1にあげられているのがおもしろい．第2に，今後の同種のライブラリ開発の基準（yardstick）を与えることを目ざしている．使われ方としては，ブラックボックス的に使われる場合と，特定の問題のためにユーザが一部手直し，または拡張して使う場合を想定している．プログラムコード，ドキュメントは明解さを十分考慮しており，大学の授業に使われることも想定している．

このため，ライブラリは的確なコメントと，構造が明確に分かるインデンテーションをもつ読みやすい FORTRAN ソースで提供されている．また，完全なマシン独立性を確保し，かつほとんどのマシン上で最適に近い性能を得ることをねらっている．

（2） 信頼性，性能の確認

上記にあげたコードのポータビリティと性能の確認は，各種の計算機をもつ 24 の大学・民間研究所で実施された．以下にその主な機関を示す．

- Argonne National Labo. (IBM 370/195)
- Bell Telephone Labo. (Honeywell 6080)
- Los Alamos Scientific Labo. (CDC 7600, CRAY-1)
- Univ. Illinois (CDC Cyber 175)

これらのテストサイトでの結果は（1）の目的を達していることを示し，さらにその実施過程でいくつかのコンパイラおよびオペレーティングシステムの潜在バグも発見したという．最近では，Alliant 社の並列計算機 Alliant FX シリーズのベンチマークとしても使われ[14)]，高い性能と信頼性を発揮している．

（3） 適用対象行列と演算の種類

LINPACK は，主として正方行列の連立一次方程式解法ルーチンから成るが，最小二乗法および統計計算関係で非正方行列（rectangular）も対象とする．また，基本的に主メモリ内(古い言い方ではイン・コア)で計算するアルゴリズムとしているため，多くの計算機で密行列では数百元，帯行列では数千元程度まで扱える．ただ，一般疎行列に対してサポートがなく，文献5) の範囲では最近のスーパコンピュータで多用される反復解法は採用されていない．

（4） LINPACK ルーチンの構成

LINPACK の1つの演算機能は通常2個のサブルーチンで処理される．1つは左辺係数行列処理用，他の1つは右辺ベクトル処理用である．これにより，係数行列が同じで，異なる右辺をもつ多数の連立一次方程式解法の処理時間短縮を可能にしている．サブルーチン群は，それが扱う行列のタイプとそこへ施す演算の種類が分かるように，次のような命名規則で名前が付けられている．すなわち，すべてのサブルーチン名は5文字で，最初の1文字は Matrix のデータタイプを表す次のいずれかである．

S : Real
D : Double Precision
C : Complex
Z : Double Precision Complex

第2，3文字は，Matrix の形または分解（decomposition）の形式を表す以下のいずれかである．

GE : General

GB : General Band
PO : Positive Definite
PP : Positive Definite Packed
SI : Symmetric Indefinite
SP : Symmetric Indefinite Packed
HI : Hermitian Indefinite
HP : Hermitian Indefinite Packed
TR : Triangular
GT : General Tridiagonal
PT : Positive Definite Tridiagonal
CH : Cholesky Decomposition
QR : Orthogonal-Triangular Decomposition
SV : Singular Value Decomposition

最後の2文字は，演算の種類を表す次のいずれかである．

FA : Factor
CO : Factor and esitimate condition
SL : Solve
DI : Determinant and/or inverse
DC : Decompose
UD : Update
DD : Downdate
EX : Exchange

LINPACK サブルーチンは，上記第2，3文字が表す Matrix データタイプと第4，5文字の表す演算の組合せの，合計48個(すべての組合せではない) ある．ユーザズガイドには，サブルーチン群ごとに概要，使用法，使用例があり，より進んだユーザのためにはアルゴリズム，プログラミング詳細情報，性能情報も提供されている．ソースプログラムリストは TAMPR というプログラムでインデンテーションが施され見やすいものである．また，たとえば倍精度複素数版サブルーチンは TAMPR によって，単精度版から自動生成されるなど工夫されている．

(5) マシン独立性

ポータビリティを高めるため，次のようにしてマシン独立性を計っている．

(a) マシン依存定数は用いない．
(b) 入出力は LINPACK 内では行わない．
(c) 文字操作はしない．
(d) Common, Equivalence 機能は使わない．
(e) 型混合の演算は行わない．

このうち，(a)に関しては，たとえば行列の singularity の判定は，マシンイプシロンといった機械依存の小さな値を用いるのでなく，行列の条件数評価によって行う．また，(e)に関しては，各 precision のルーチン内では，それ以上の拡張精度演算はいっさい行わない．したがって LINPACK ルーチンとしては，反復改良法 (Iterative Improvement) を行っていない．これを行うには，残差計算に，より高い精度が要求されるからである．

(6) 精度に対する配慮

計算結果として最終的に高い精度を得るため，LINPACK では特にスケーリングを重視している．すなわち，もとの連立一次方程式

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b} \qquad ①$$

は一般に次のようにスケーリングされて解かれる．

$$(D_r A D_c)\,(D_c^{-1}\boldsymbol{x})=(D_r\boldsymbol{b}) \qquad ②$$

ここで D_r は行スケーリング用対角行列，D_c は列スケーリング用対角行列である．

方程式①と②は数学的には等しいが，丸め誤差のため①と②では同じ解法で全く解の精度が異なる場合があることは良く知られた事実である．

スケーリングの必要性は，行列 A に関する種々の初期誤差により生じている．初期誤差はまず，A の要素の測定誤差（なんらかの測定値であれば）や，内部数値表現にともなう丸めが原因である．また要素値がなんらかの計算結果として得られるものなら，打切り誤差と丸め誤差が含まれている．このような誤差を表す誤差行列 E が $A+E$ を singular にするほど大きければ，LINPACK ルーチンの計算はほとんど無意味になる．一方 E が微小であっても，A と $A+E$ に対する数値解が大幅に異なる場合がある．そこで，A がどれほど singular に近いか，また，E がどの程度数値解を変動させるかを判断する手段が必要になる．その1つは行列の条件数の評価である．LINPACK では，行列 A の条件数 $K(A)$ を次のように見積もることを可能にしている．

$$\frac{1}{f_n K(A)}\leq\frac{\nu(E)}{\nu(A)}\leq\frac{f_n}{K(A)}$$

ここで E は，$A+E$ を singular にする最小の誤差行列であり，$\nu(A)=\max_{i,j}(|a_{ij}|)$ である．f_n は条件数評価ルーチンで使われる係数である．

詳しいことは略すが，上記の条件数評価法から，誤差行列 E の各要素がほぼ同程度の大きさでなければ，$K(A)$ に意味がなくなる．上に述べたスケーリングは，このようなことがなくなるようにし，LINPACK での条件数評価が意味のあるものにするためのもので

ある．この実現方法は基本的にはユーザにまかされた形になっているが，いくつかの指針は，ユーザガイドにも述べられている．

3.2 偏微分ソルバ

スキーム化を含めてプログラミングを支援する偏微分ソルバには，(米) Purdue 大学の ELLPACK[6]，(米) IMSL 社の PDE/PROTRAN®[15]，(独) Karlsruhe 大学の FIDISOL[16] などがあるが，ここでは日立の DEQSOL[7),11),12)]を紹介することにする．

DEQSOL (Differential EQuation SOLver Language) は数値シミュレーション向きに設計した高水準のプログラミング言語で，偏微分方程式系から成る問題の解法を数値計算アルゴリズムのレベルで記述することができる．DEQSOL によって記述されたプログラムは，DEQSOL トランスレータにより FORTRAN のシミュレーションコードに自動変換される．

本システムの目的は，第1にプログラムの生産性を飛躍的に向上させること，第2に DEQSOL プログラム中に内包されるアルゴリズムの並列性を利用して演算並列化率（ベクトル化率）の高いシミュレーションコードを自動生成することにある．以下に DEQSOL の概要，コードの自動生成方式，評価などについて述べる．

3.2.1 DEQSOL の概要

簡単な記述例を用いて，言語の概要を説明する．

初期温度 100℃ のボックスを，左右の壁は断熱，下部から 200℃ で加熱し，上部より熱の出入りを許した際の内部の温度変化を求めるモデルと，その DEQSOL による記述例を図-5 に示す．DEQSOL のプログラムは，数理モデルの構造を記述する部分と，計算スキーム(アルゴリズム)を記述する部分の2つに大きく分けられる．ステートメント(3)から(10)まではこの前者にあたる．生成するプログラム名(1)と離散化手法(2)の指定に続いて，問題の定義されている空間領域(3)，非定常問題に対する時間領域(4)，差分法による計算格子(5)，未知物理変数(6)，物理定数(7)，各種条件や局所的な値の設定時に引用する副領域(8)初期条件(9)，境界条件(10)などを指定する．次に SCHEME 文(11)から END SCHEME 文(17)の間で，計算スキーム（アルゴリズム）を記述している．本例における計算スキームは，時間微分を陽的もしくは陰的のいずれかで評価することにより得られる．図-5 の記述例はオイラ型の陽解法スキームの例で，ITERATION 文(12)から END ITERATION 文(16)の間の実行文は，UNTIL 以下の条件が成立するまで実行される．代入文(14)により各時間ステップの温度が順次求められる．仮に陰解法，たとえばバックワードオイラ法を構成するには，代入文(14)を SOLVE 文(19)で置きかえれば良い．SOLVE 文は DEQSOL の基本機能の1つで，任意の線型偏微分方程式中の指定した変数の数値解を，BY 以降で指定した行列解法によって求めることを指定する．

以上のように DEQSOL 言語で記述されたプログラムは，DEQSOL トランスレータによって，図-6 に示す手順で FORTRAN のシミュレーションコードに自動変換され，翻訳・連結・実行後，数値結果はポストプロセッサ SGRAF によって，グラフ・画像などで表示される．

現在の DEQSOL の適用範囲は，1～3 次元領域上で定義された2階までの偏微分方程式系で，数値解析手法として差分法と有限要素法の両手法を実現している．言語仕様の詳細や他のスキーム記述例などについては参考文献を参照願いたい．

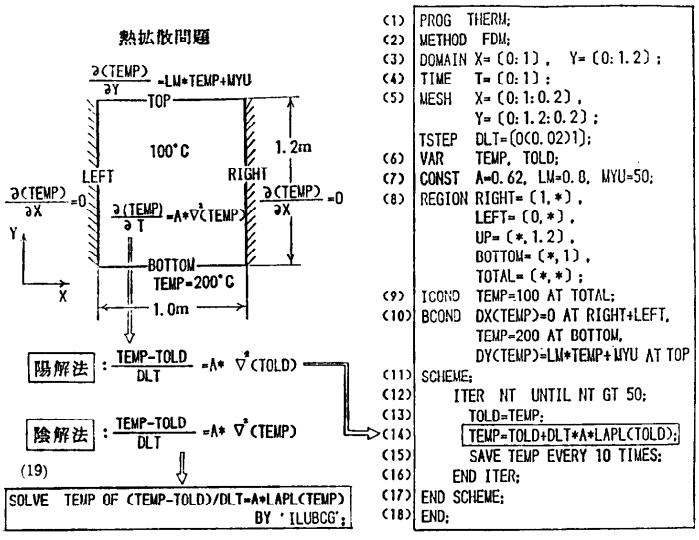
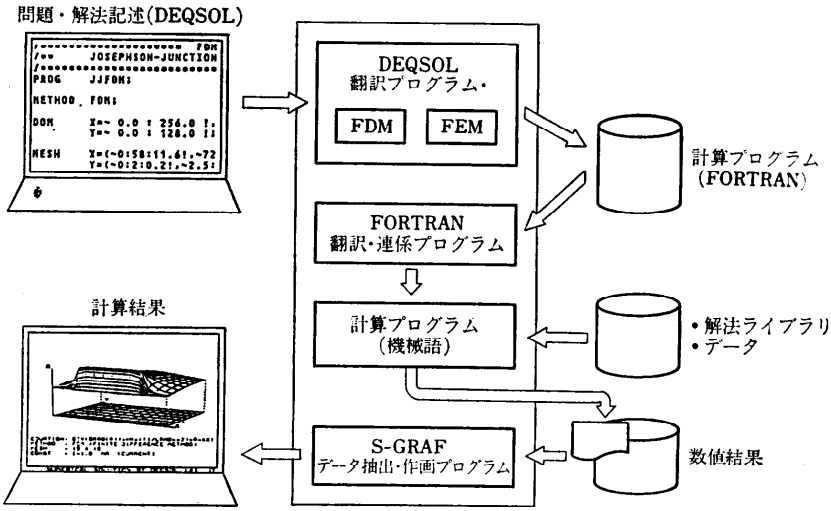


図-5 DEQSOL の記述例

3.2.2 コード生成方式

DEQSOL トランスレータはパイプライン方式のベクトルプロセッサに対して効率の良い計算コードを自動生成する．代入文と SOLVE 文



DEQSOL : Differential EQuation SOLver　　FDM : Finite Difference Method
S-GRAF : Scientific GRAphing Facilities　　FEM : Finite Element Method

図-6 DEQSOL のシステム構成

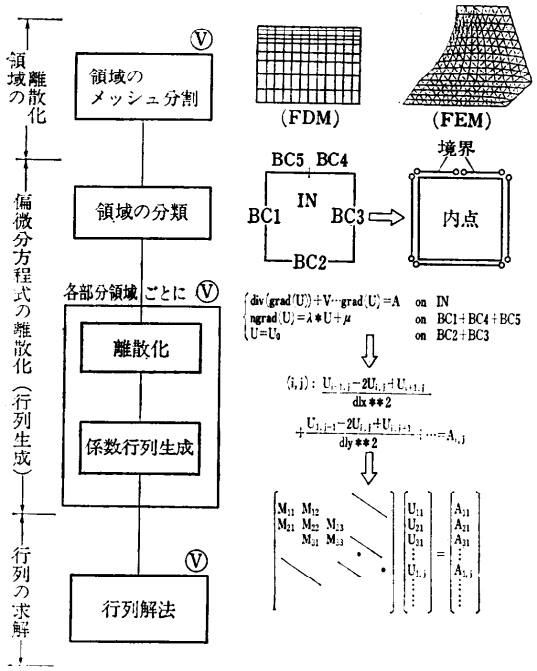


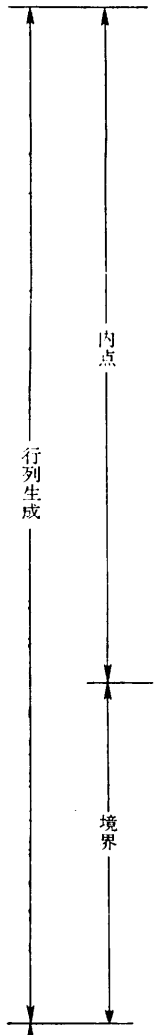
図-7 DEQSOL における処理の流れ

の自動離散化処理が中心的な役割を占めるが，離散化手法として差分法が指定されたときは 1/2 中心差分による離散化規則を微分演算子に階層的に施すことによりコントロールボリューム法と同等の離散化を実現している．有限要素法が指定されたときはガレルキン法により離散化する．基本となる技術は，離散化規則に従って微分式を展開する数式処理と，空間領域を適用する離散化規則により分割する領域分割処理である．以下では特に差分法における SOLVE 文のコード生成について述べる．

図-7 は SOLVE 文に対するコード生成フローを示している．トランスレータは初めに，偏微分方程式と境界条件式および離散化規則に従って，領域を同一の式と離散化規則に支配される部分領域に分割する．各部分領域はその内部の各格子点（節点）において，同一形式の離散式が生成される最大の単位となる．換言すれば各部分領域がその計算処理を単一の DO ループで構成可能な最大の単位となる．たとえば 5 種類の境界条件が与えられている **図-7** の例では，1 つの内点領域と 5 つの境界領域からなる計 6 つの部分領域に直和分割される．ついで各部分領域ごとに，離散化を行って離散式を得て係数成分を抽出し，連立一次方程式系の係数行列と定数ベクトルのその部分領域の該当する部分の値を計算する FORTRAN のコードを生成する．すべての部分領域の係数計算コードの生成後，すなわち行列生成の完了後，あらかじめ備えているベクトルチューニングされた行列解法ライブラリをリンクする CALL 文を生成する．

図-8 にある SOLVE 文に対する自動生成コードの

```
SOLVE DN OF (NNEW*(DN-N)/DLT-1/2*(DIV(D*GRAD(NNEW))
            +DIV(DP*DN*GRAD(NNEW)))+DIV(D*GRAD(DN))
            -DIV(DO*GRAD(N)))                    BY 'ILUCR';
```




```
      DO 47 J=IDXP3,IDXP8
      DO 47 I=IDXP3,IDXP8
      ¥CONS(I+IDXP10*J)=-(-(NNEW(I,J)/DLT)+N(I,J)/DLT+1.0/2.0*(((D(I+1,J
     &)+D(I,J))/2.0)*((NNEW(I+1,J)-NNEW(I,J))/DLX(I+1))-((D(I,J)+D(I-1,J
     &))/2.0)*((NNEW(I,J)-NNEW(I-1,J))/DLX(I)))/((DLX(I+1)+DLX(I))/2.0)+
     &1.0/2.0*(((D(I,J+1)+D(I,J))/2.0)*((NNEW(I,J+1)-NNEW(I,J))/DLY(J+1)
     &)-((D(I,J)+D(I,J-1))/2.0)*((NNEW(I,J)-NNEW(I,J-1))/DLY(J)))/((DLY(
     &J+1)+DLY(J))/2.0)+1.0/2.0*(((DO(I+1,J)+DO(I,J))/2.0)*((N(I+1,J)-N(
     &I,J))/DLX(I+1))-((DO(I,J)+DO(I-1,J))/2.0)*((N(I,J)-N(I-1,J))/DLX(I
     &)))/((DLX(I+1)+DLX(I))/2.0)+1.0/2.0*(((DO(I,J+1)+DO(I,J))/2.0)*((N
     &(I,J+1)-N(I,J))/DLY(J+1))-((DO(I,J)+DO(I,J-1))/2.0)*((N(I,J)-N(I,J
     &-1))/DLY(J)))/((DLY(J+1)+DLY(J))/2.0))
      ¥¥NAME(I+IDXP10*J)=-(1.0/DLT)+1.0/2.0*(DP(I+1,J)+DP(I,J))/2.0/2.0*
     &(NNEW(I+1,J)-NNEW(I,J))/DLX(I+1)/((DLX(I+1)+DLX(I))/2.0)-(1.0/2.0*
     &(DP(I,J)+DP(I-1,J))/2.0/2.0*(NNEW(I,J)-NNEW(I-1,J))/DLX(I)/((DLX(I
     &+1)+DLX(I))/2.0))+1.0/2.0*(DP(I,J+1)+DP(I,J))/2.0/2.0*(NNEW(I,J+1)
     &-NNEW(I,J))/DLY(J+1)/((DLY(J+1)+DLY(J))/2.0)-(1.0/2.0*(DP(I,J)+DP(
     &I,J-1))/2.0/2.0*(NNEW(I,J)-NNEW(I,J-1))/DLY(J)/((DLY(J+1)+DLY(J))/
     &2.0))-(1.0/2.0*(D(I+1,J)+D(I,J))/2.0/DLX(I+1)/((DLX(I+1)+DLX(I))/2
     &.0))-(1.0/2.0*(D(I,J)+D(I-1,J))/2.0/DLX(I)/((DLX(I+1)+DLX(I))/2.0)
     &)-(1.0/2.0*(D(I,J+1)+D(I,J))/2.0/DLY(J+1)/((DLY(J+1)+DLY(J))/2.0))
     &-(1.0/2.0*(D(I,J)+D(I,J-1))/2.0/DLY(J)/((DLY(J+1)+DLY(J))/2.0))
      ¥NAMEX(I+IDXP10*J)=-(1.0/2.0*(DP(I,J)+DP(I-1,J))/2.0/2.0*(NNEW(I,J
     &)-NNEW(I-1,J))/DLX(I)/((DLX(I+1)+DLX(I))/2.0))+1.0/2.0*(D(I,J)+D(I
     &-1,J))/2.0/DLX(I)/((DLX(I+1)+DLX(I))/2.0)
      ¥¥NMXP(I+IDXP10*J)=+1.0/2.0*(DP(I+1,J)+DP(I,J))/2.0/2.0*(NNEW(I+1,
     &J)-NNEW(I,J))/DLX(I+1)/((DLX(I+1)+DLX(I))/2.0)+1.0/2.0*(D(I+1,J)+D
     &(I,J))/2.0/DLX(I+1)/((DLX(I+1)+DLX(I))/2.0)
      ¥NAMEY(I+IDXP10*J)=-(1.0/2.0*(DP(I,J)+DP(I,J-1))/2.0/2.0*(NNEW(I,J
     &)-NNEW(I,J-1))/DLY(J)/((DLY(J+1)+DLY(J))/2.0))+1.0/2.0*(D(I,J)+D(I
     &,J-1))/2.0/DLY(J)/((DLY(J+1)+DLY(J))/2.0)
      ¥¥NMYP(I+IDXP10*J)=+1.0/2.0*(DP(I,J+1)+DP(I,J))/2.0/2.0*(NNEW(I,J+
     &1)-NNEW(I,J))/DLY(J+1)/((DLY(J+1)+DLY(J))/2.0)+1.0/2.0*(D(I,J+1)+D
     &(I,J))/2.0/DLY(J+1)/((DLY(J+1)+DLY(J))/2.0)
47    CONTINUE
      J=IDXP9
      DO 48 I=IDXP3,IDXP8
      ¥CONS(I+IDXP10*J)=-(-(NNEW(I,J)/DLT)+N(I,J)/DLT+1.0/2.0*(((D(I+1,J
     &)+D(I,J))/2.0)*((NNEW(I+1,J)-NNEW(I,J))/DLX(I+1))-((D(I,J)+D(I-1,J
     &))/2.0)*((NNEW(I,J)-NNEW(I-1,J))/DLX(I)))/((DLX(I+1)+DLX(I))/2.0)+
     &1.0/2.0*(((D(I,J+1)+D(I,J))/2.0)*((NNEW(I,J+1)-NNEW(I,J))/DLY(J+1)
     &)-((D(I,J)+D(I,J-1))/2.0)*((NNEW(I,J)-NNEW(I,J-1))/DLY(J)))/((DLY(
     &J+1)+DLY(J))/2.0)+1.0/2.0*(DP(I,J+1)+DP(I,J))/2.0*2.0*DLY(J)*(0.0)
     &/2.0*(NNEW(I,J+1)-NNEW(I,J))/DLY(J+1)/((DLY(J+1)+DLY(J))/2.0)+1.0/
     &2.0*(D(I,J+1)+D(I,J))/2.0*2.0*DLY(J)*(0.0)/DLY(J+1)/((DLY(J+1)+DLY
     &(J))/2.0)+1.0/2.0*(((DO(I+1,J)+DO(I,J))/2.0)*((N(I+1,J)-N(I,J))/DL
     &X(I+1))-((DO(I,J)+DO(I-1,J))/2.0)*((N(I,J)-N(I-1,J))/DLX(I)))/((DL
     &X(I+1)+DLX(I))/2.0))
      :
      DO 52 J=IDXP2,IDXP9
      DO 52 I=IDXP2,IDXP9
      ¥¥ANS(I+IDXP10*J)=DN(I,J)
52    CONTINUE
      CALL DILUCR(2,IDXP9,IDXP9,IDXP1,¥¥NAME,¥¥NMXP,¥NAMEX,¥¥NMYP,¥NAMEY
     &,¥¥NMXP,¥NAMEX,¥CONS,¥¥ANS,¥¥A,¥¥B,¥¥C,¥¥D,¥¥C1,¥¥D1,¥¥A1,¥¥R,¥¥P,
     &¥¥Q,¥¥01,¥¥W,1.0E-5,IERR)
      IF (IERR.NE.0) WRITE(6,*) ' RETURN-CODE = ',IERR
      DO 53 J=IDXP2,IDXP9
      DO 53 I=IDXP2,IDXP9
      DN(I,J)=¥¥ANS(I+IDXP10*J)
53    CONTINUE
```

図-8　DEQSOL で自動生成された FORTRAN の構成

表-1 DEQSOL の性能評価

	プログラム名		CVD	PCAD	WEL	JJD	MHD	EBT
問題	離散化手法(次元)		FDM(3)	FDM(2)	FDM(2)	FEM(2)	FEM(3)	FEM(2)
	問題規模	メッシュ数	10*10*10	25*25	99*99	600	23,040	2,215
		節点数	1,331	676	10,000	341	4,725	1,245
効果	コード行数	DEQSOL	127	79	67	96	904	239
		自動生成FORTRAN	1,361	1,312	1,091	1,078	11,558	2,476
		生成効率	(10.7)	(16.6)	(16.3)	(11.2)	(12.8)	(10.4)
	ベクトル化率(S-810)〔%〕		91.0	96.4	94.1	92.7	93.9	93.7
	加速率(S-810)		3.8	8.2	9.4	5.2	7.7	5.5
処理性能	翻訳リージョンサイズ〔M bytes〕		3	3	3	3	5	3
	翻訳 CPU タイム〔on M 200 H・秒〕		266.5	37.6	17.4	9.4	367.1	25.1

CVD: 化学蒸着における気流解析　　JJD: JJ 素子の磁場ポテンシャル解析
PCAD: LSI プロセスにおける不純物拡散　　MHD: 磁気ディスクの磁界解析
WEL: LSI WEL 層のデバイス解析　　EBT: EB 装置の電位分布解析

一部を示す．上述したように，係数行列計算は部分領域単位の DO ループによって構成されている．

ここでは主に差分法のコード生成を例にとって述べたが，有限要素法でも基本的な考え方は同様で，ガレルキン法によって得られた領域積分項の全体行列への寄与と各境界積分項の寄与を，全領域と境界部分領域単位の DO ループによって計算するコードを生成する．

3.2.3 評　価

表-1 に DEQSOL の実問題への適用事例の一部とそれを通して得られた評価データを示す．本表は各適用例題に対して，離散化手法，問題規模，DEQSOL の記述効率，生成コードのベクトル化率とベクトル計算時のスカラ計算に対する性能向上比（加速率），および DEQSOL トランスレータの処理性能を示している．

これらの評価から以下のことが分かる．

（a） プログラムの生産性は記述行数を尺度として FORTRAN に比べて一桁程度高い．

（b） 自動生成コードは HITAC S-810/20 において，90% 以上の高いベクトル化率を達成している．

（c） DEQSOL トランスレータのコードの自動生成負荷は，作業領域で 3〜5 MB，処理性能として HITAC M-200 H で数秒から数分程度と比較的軽い．

3.3 統合問題解決支援環境

統合問題解決支援環境（PSE）の一例として，米国 ORNL (Oak Ridge National Laboratory) の NIT-PACK に端を発し，現在ノルウェイの CMI (Chr. Michelsen Institute) で開発が進められている NEXUS[10] の概要を紹介する．

NEXUS は，グラフィックインタフェース，分野ごとの自然でかつ効率の良い解法探索ならびに知識ベース，プログラムベースの提供を特徴としている．

解法探索は，分野ごとに用意された探索トリーの各ノードでユーザが自然語で表示されるメニュを選択して探索パスを限定することにより進めるが，特に探索トリー上の “analysis point” において与えられた問題の数理的な性質を自動的に解析し，その結果に基づく推論と，パスの自動限定などを行うので正確かつ容易な探索ができるという．また解析と探索，複数パスの探索はマルチタスク機能ないし並列プロセッサを用いて並列に行うことが計画されている．

現在，他機関の協力のもとに知識ベースの整備が進められている分野は，固有値・固有ベクトルを含む線型計算，常微分方程式，多項式のゼロ点，特殊関数の近似計算などであり，特に常微分方程式の解法について検討が進んでいる．この分野では，探索の結果に基づいて計算コードを自動生成するためのプログラムベースの整備も進められている．

以上 NEXUS の概要を紹介したが，このほか，米国 Purdue 大学の著名な偏微分ソルバ ELLPACK も最近 PSE としての性格を強めている[9]．

4. むすび

以上，数値シミュレーションのための自動プログラミングツールを性格付けし，性格の異なる3つの事例を紹介した．技術計算における自動プログラミングの困難さは，多岐にわたる数理面のノウハウを反映できる仕掛を残さなければならない点にあり，そのため従来は限られた機能ソフトウェアを除いて，自動化の手段が分野別のパッケージソフトとプログラミング言語に二極化する傾向にあったが，最近ようやく高水準言語のアプローチによりスキームのレベルまで自動化の試みがなされるようになった．しかしこのレベルで利用者に不便を感じさせないためには対話機能が不可欠に思える．その意味で，自動化ツールは今後ますます統合問題解決支援環境としての性格を強めてゆくと予想される．そこでは個別の技法の充実とともに，それらの間の有機的な結合やマンマシン性，知識処理の援用などが求められてゆくであろう．

参考文献

1) 上村他：計算理学，数理科学，サイエンス社 (1985.10).
2) 大特集：数式処理，情報処理，Vol. 27, No. 4 (1986).
3) 日立製作所：数値計算副プログラムライブラリ MSL-Ⅱ.
4) 二宮市三，秦野甯世：数学ライブラリ NUMPAC，情報処理，Vol. 26, No. 9, pp. 1033-1042 (1985).
5) Dongarra, J. J. et al.: LINPACK User's Guide, SIAM, Philadelphia (1979).
6) Rice, J. R. and Boisvert, R. F.: Solving Elliptic Problem Using ELLPACK, CSD-TR 414, Computer Science Department, Purdue University, Sept. 1982 (Revised May 1983).
7) 梅谷他：数値シミュレーション用プログラミング言語 DEQSOL，情報処理学会論文誌，Vol 26, No 1, pp. 168-180 (1985).
8) Ford, B. and Iles, R. M.: The What and Why of Problem Solving Environments for Scientific Computing, IFIP WG 2.5 Working Conference 4, Problem Solving Environments for Scientific Computing, North-Holland (1987).
9) Rice, J. R.: ELLPACK: An Evolving Problem Solving Environment, 同上 (1987).
10) Gaffney, P. W. et al.: NEXUS: Towards a Problem Solving Environment (PSE) for Scientific Computing, ACM SIGNUM Newsletter, Vol. 21, No. 3, pp. 13-24 (July 1986).
11) Konno, C. and Umetani, Y.: A High Level Programming Language for Numerical Simulation: DEQSOL, Denshi Tokyo No. 25, IEEE Tokyo Section, pp. 50-53 (1986).
12) Konno, C. et al.: Advanced Implicit Solution Function of DEQSOL and its Evaluation, Proc. of EJCC, pp. 1026-1033 (Nov. 1986).
13) 小国 力：FORTRAN 77-応用ソフトウェア作成技法，丸善(株) (昭和61年10月).
14) Perrenod, S. C.: Parallelism Made Easy to Use: The Alliant FX Series Minisupercomputer, 第3回大型行列計算シンポジウム配布資料 (1987).
15) PROTRAN®: A Comprehensive Problem-Solving Environment, IMSL Inc.
16) Schönauer, W. et al.: PDE Software for Vector Computers, in Advances in Computer Methods for Partial Differential Equations, IMACS, pp. 258-267 (1984).
17) 菅 忠義：FORTRAN 8X の紹介，bit, Vol. 17, No. 12.

(昭和62年8月17日受付)