ハイパフォーマンス
コンピューティング 53－1
（１９９４．１０．２７）

数値シミュレーションプログラム生成支援に関する研究
－差分法プログラム生成支援－

Choompol Boonmee、真鍋保彦、渋井俊昭、釣谷浩之、森正孝、川田重夫
長岡技術科学大学

大規模の数値シミュレーションプログラムの作成は一般的に膨大な労力と時間を必要としている。その労力と時間を削減するために様々な支援システムが研究・開発されてきた。しかしそれらのシステムは、数値シミュレーションの知識を持ち正確さを求める技術者や他のユーザにとっても幾つかの解決すべき問題がある。ユーザとの不十分なインタラクション、生成されたソースコードの検証・修正が行えないなどが挙げられる。これらの問題を解決し技術者が求める実用的な支援環境の実現の可能性を探る。

Problem-Solving Environment for Partial-Differential Equations, Based on Finite Difference Method

Choompol Boonmee, Yasuhiko Manabe, Toshiaki Shibui, Masataka Mori, and Shigeo Kawata
Nagaoka University of Technology,
1603-1 Kamitomioka, Nagaoka, Niigata. 940-21, Japan.

Great efforts and a long work are required in order to produce a large-scale numerical simulation source code. So far several studies have been performed to ease the hard work; Recently problem-solving environments(PSE) have been intensively studied in various fields. However the PSE systems developed may have the following problems which should be studied and solved; An insufficient interaction between the system and a user, and a lack of the verification method for the correctness of the results obtained by the system. In this paper we present a study on our PSE which solves the problems.

1.導入

一般に数値シミュレーションといえば様々なものが考えられる。例えば、物理分野でよく知られている偏微分方程式(Partial Differential Equations)を解くようなPDE問題がある。他にも化学分野のシミュレーションや生物学分野のシミュレーションや経済学分野でのシミュレーションなどがある。

ここではPDE問題の数値シミュレーションに注目する。PDE問題を解く方法としてよく知られている数多くの手法が提案されてきた。大きく分類すれば差分法と有限要素法が挙げられる。本研究では前者の差分法に注目している。差分法の中でも様々な手法が提案されている。その代表的なものとしては単段階陽解法、多段階陽解法、陰解法、反復法などが挙げられる。物理モデルと数学モデルがユーザにより決定された後、どの解法を用いても次のような段階を踏まえて行うであろう。

0.計算空間・偏微分方程式・境界条件を与える。(PDEs & BCs)

1.計算空間の格子構造のモデル化(Grid Structure Modeling)

2.適切な解法の選択(Selection of Solver Method)

3.偏微分方程式の離散化（差分化=Discretiazation）・差分式の変形

4.差分式を解くためのプログラム設計・作成(Program Generation)

5.プログラムを実行し、数値結果を得て、可視化する。

上述したように数値シミュレーションプログラムの作成に必要な作業と流れを図化すると次のようなものになる。

図1.数値シミュレーションプログラムの作成作業の流れ

数値シミュレーションプログラムを作成する度にこの一連の作業をしなければならない。しかし特に１番の格子構造のモデル化から４番目のプログラム作成までの作業はある程度の一般性を持ち、確立されている知識（解法についての知識）に従って機械的にできる作業である。それにもかかわらず、数値シミュレーションにおいては計算モデルの修正や精密化によってプログラムの改良や更新を行っていくことが多いが、これに必要な工数・時間は一般的に膨大となる。

本研究ではこの一連の作業を、人間の得意なところを人間に任せ、コンピュータが得意な機械的な作業をコンピュータに任せられるような実用的なPDE問題解決環境の実現の可能性を探る。本研究の目的としては、上述した数値シミュレーションプログラム作成に必要とした作業をいかにしてコンピュータにさせることができるかを研究することである。

従来、同様の目的で研究開発されてきたシステムとして、DEQSOL[1-4]、ELLPACK[5]、ALPAL[6]、EVE[7]などといったシステムが挙げられる。システムの実現方式は大きく分け

ると数値計算用のソースプログラムを生成するものと多数のライブラリを揃えてシステムからそれらをドライブするライブラリ型がある。上述した例の中ではDEQSOL、ALPALなどは前者に、ELLPACK、EVEなどは後者に属する。しかし、シミュレーションの知識を持ち正確さを求める技術者にとって従来のシステムには以下の様な幾つかの解決すべき問題がある。

1. ライブラリ型システムでは、ユーザは実際計算に使われているプログラムがいかなるものかも分からないし、そのソースプログラムを直接検証・修正することもできない。
2. ソースプログラムを提供するシステムでは、生成されたプログラムは人間向けの（人間が読めるような）プログラムではないのでユーザがソースプログラムを直接検証・修正・再利用することが困難である。
3. 多くのシステムでは人間的に（人間が行う様に）問題解決の作業を進めていくのではなく、標準パターンに当てはめて機械的に置換えていく方法で実現している。人間にとっては分かりにくい。
4. ユーザとの対話、ユーザとの情報交換が不十分である。

2.システム概要

我々は次のような特徴を持つ（PDE問題）数値シミュレーションプログラム生成支援環境(NCAS)に関する研究・開発を行ってきた[8,9]。

1. 数値シミュレーションプログラムを作成する過程を人間と対話しながら人間的に作業を進めていくユーザ・インタラクティブな環境。
2. 人間的な可読性の高い理解しやすい数値シミュレーションプログラムを生成する。すると、ユーザがプログラムを直接検証・修正・再利用することが容易になる。
3. システムがPDE問題を解く標準的な解法の知識を持ち、ユーザがそれらを選択して利用できる

本研究の目指すシミュレーション支援システムは、必要な諸情報を入力するとただプログラムを生成するだけのパッケージのようなものではなく、人間のパートナーのように人間と対話しながら作業を進める環境すなわちユーザ・インタラクティブな環境を装備するものである。

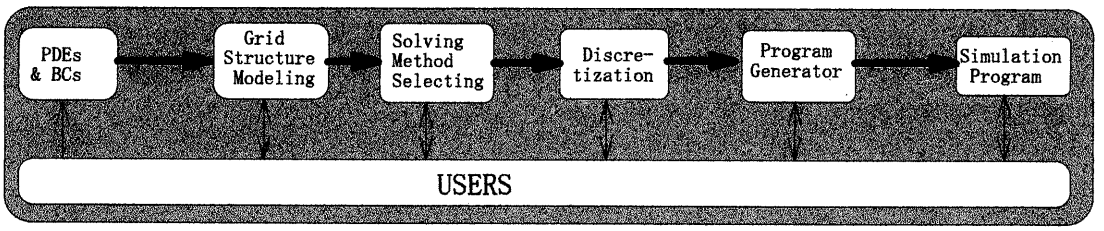

図2.ユーザ・インタラクティブな環境

2.1 ユーザとのInteraction

ユーザとのInteractionが必要な理由を述べる。

1. ユーザが望ましいようなプログラムを生成するためにユーザの要求・情報が必要である。
2. ユーザの要求を正確にシステムに伝えられるようにユーザとシステムとのお互いの 正確な理解が求められる。そのためにユーザとシステムのInteractionが必要である。

又効率のよいInteractionを実現するには何が必要かについて述べる。

1. 視覚的に分かりやすいユーザ・インターフェースが好ましい。それにはGraphical User Interface(GUI)が一番適切であろう。
2. 人間が行うような手順で作業を進めることが望ましい。
3. 作業の途中経過をユーザに知らせる必要がある。
4. 人間に判断させるべきなところは人間に聞く。

2.2 グリッド構造のモデル化

まず、計算に用いるグリッド構造がどういったものか、均一メッシュか、何分割するか、外部からのメッシュ情報を取り込むか等の情報をユーザが自由に決定できるようにする必要がある。また計算するその物理量をメッシュ上の定義場所を明示的にディスプレイに表示し、そしてユーザがその定義場所を自由に決定できるようにする必要がある。メッシュ上の定義できる場所としては次の図で示すようなものである。

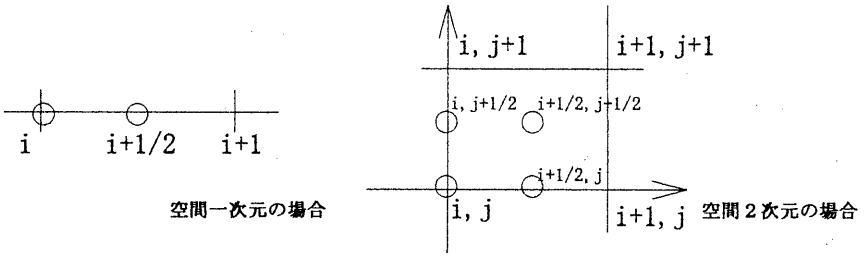

図3. 物理変数の場所の定義

又，いくつのどういった境界があるかユーザが確認・設定できるようなインターフェースが必要になる。以上のような機能を実現すると次の図で示したような画面が考えられる。この画面は本システムにおける一提案を実現したものである。グリッドの寸法、物理量とその定義場所が表示されている。

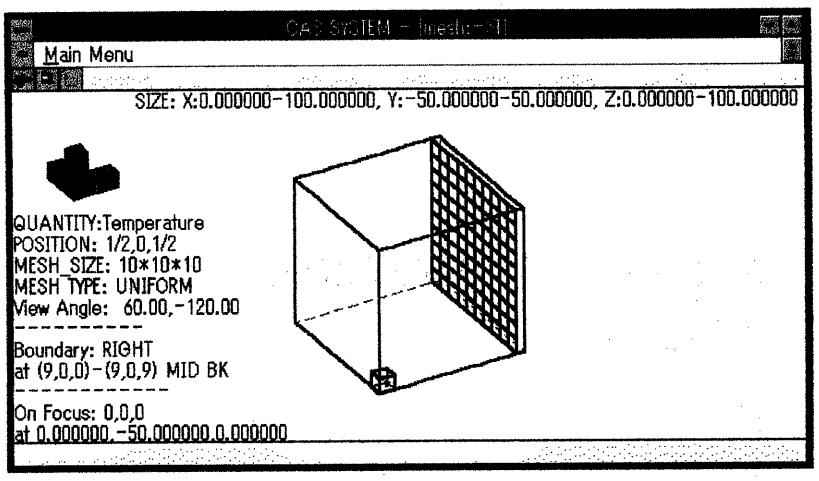

図4. グリッド構造のモデル化のウィンドウでグリッド構造についての情報が３次元的に表示される.

2.3 適用すべき解法の選択

本研究ではPDE問題を解く方法として差分法を対象としている。問題の種類・性質によって適用できる解法、あるいはその問題に適切な解法が違ってくる。人間がその問題に適すると思われる解法を選択する。差分法の中でも良く知られる様々な解法がある。代表的なものとして陽解法、陰解法、反復法等がある。人間がPDE問題を分類しどの解法が一番適切かを選択する。この過程を次の図で示している。

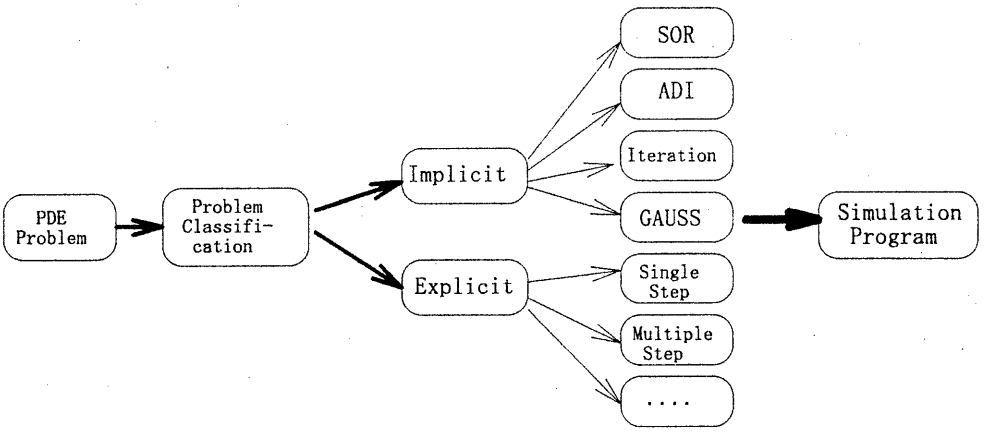

図5. Selection of PDE-Problem-Solving Method

解法を選択すると次の段階へと進む。次の段階としては選択された解法の手順（知識）に従って離散化し、差分式を変形して、プログラム生成を行う。

2.4 離散化

解法が決まるとその解法に合うように離散化し、計算式を生成できるように項のまとめ・

項の移動を行い、離散化された差分式を変形する。理解やすいプログラムにするため、項のまとめ、変数置換を行う。差分化・変形の過程もユーザが納得できるようにユーザ・インターフェースを通じて表示する。差分化を行う際にユーザに細かい指示が必要な場合（風上差分のような特別な差分化を行いたい場合等）があるのでユーザからの指示を受けられるようにする。本システムでは次のような画面で実現している。

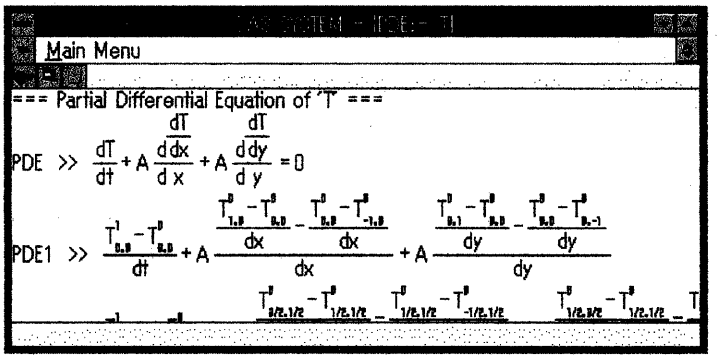

図6. 差分化及び差分式の変形のウィンドウ

2.5 プログラムの生成

可読性の高いプログラムがなぜ必要であるかについて考察する．プログラムの可読性が高いと次のような大きなメリットがあるからである。

1. ユーザが計算プログラムを直接検証・修正したい場合がある。読みやすいプログラムであればプログラムを容易に理解でき、ユーザが直接プログラムの検証・修正することができる。
2. 一部のパラメータを書き換えてパラメータスタディなどにプログラムを直接変更することができて、プログラムの再利用ができる。

次に可読性の高いプログラムを生成するには何が必要かについて考える．可読性の高いプログラムとは何か様々な解釈ができると思われるがここでは可読性の高いプログラムとは次のように考える。1)モジュール化されたプログラム、つまり一まとまりの仕事を極力サブルーチン化する。2)GOTO文を極力さける。3)長い計算式を適切な位置で打ち切って分かりやすい形にする。4)コメント文を記述する。3)の例として次の２次元拡散方程式考えてみよう。$\frac{\partial T}{\partial t}+\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0$、これを差分化して以下の様にする。

```
 TEMP(I,J)=TOLD(I,J)+DT*DX**-2.D0*TOLD(I+1,J)+DT*2.D0*DX**-2.D0*
&TOLD(I,J)+DT*DX**-2.D0*TOLD(I-1,J)+DT*DY**-2.D0*TOLD(I,J+1)-DT*
&2.D0*DY**-2.D0*T(I,J)+DT*DY**-2.D0*T(I,J-1)
```

これは読みにくいので分かりやすく書き換えると次のようなものが考えられる。

```
 TEMP(I,J) = TOLD(I,J)
&          + DT*DX**-2.D0 * (TOLD(I+1,J) - 2.D0*TOLD(I,J) + TOLD(I-1,J))
```

```
&        + DT*DY**-2.D0 * (TOLD(I,J+1) - 2.D0*TOLD(I,J) + TOLD(I,J-1))
```

3. システム構成及びプラットフォーム

本システムNCASでは、数値シミュレーションプログラムを作成するための諸情報を入力仕様ファイル(Input Specification File)として保有し、このファイルに入っている情報（解くべき物理量、偏微分方程式群、境界条件等）に基づいてユーザと対話しながらプログラム生成まで作業を逐次に行う。本システム構成は次のように図化できる。

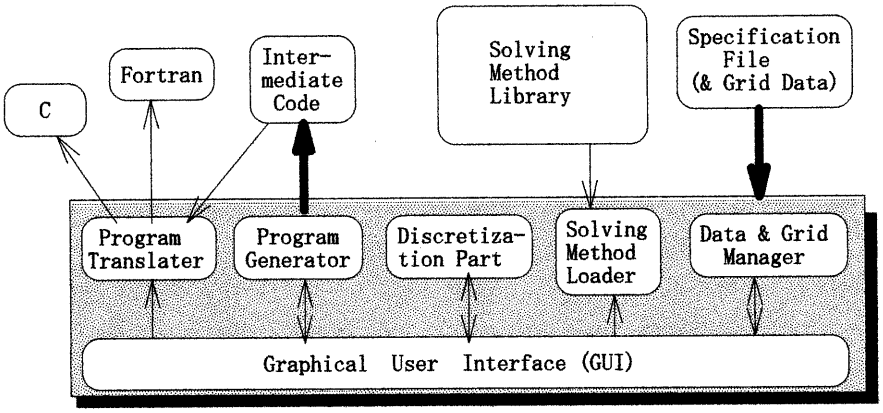

図7. NCAS System Structure

まず、システムはユーザが指定した入力仕様ファイル(Input Specification File) から諸情報を読み込む。それらの情報をユーザ・インターフェース部(Graphical User Interface) を通してユーザに表示する。グリッド形状、物理量、解法等をユーザに確認・修正させる。解法が決まるとその解法に関する知識（問題を解くための手順を記述した知識）を知識ローダ部(Solving Method Loader)でSolving Method Libraryからシステムにロードする。その手順に従って離散化部 (Discretization Part)で離散化を行い、差分式を計算できるような適切な形に変形する。解法の知識に基づいてプログラム生成部(Program Generator)でプログラム生成を行う。最初に生成されるプログラム言語はCやFortranのような汎用の高級言語ではなく、本研究で独自に設計した中間言語(Intermediate Language)である。その後プログラム変換部(Program Translater)でその中間言語をCやFortran言語に変換する。

本システムはユーザと効率よく対話できるためGUIを必要としている。また偏微分方程式の差分化や式の変形や知識の扱いなどといった記号処理的な機能も必要としている。従って、本システムの開発プラットフォームとしてはグラフィックス機能と記号処理機能の両方の機能を持つものがものが望ましい。グラフィックス機能ではCを、記号処理ではLISP言語を用いることにした。両方の言語を使えるような開発プラットフォームを設計・構築を行った。

4. まとめ・今後の課題

現在本システムを開発途中であり、まだ完成はしていないが、本研究の目指しているシステムは十分実現可能であると考えられる。又，現在のシステムではユーザに適用する解法を選択してもらうことになっているが、実は偏微分方程式を記号処理的に解析すればどういった解法が適切かをある程度判断できるのでユーザにアドバイスするあるいは自動的に解法を選択し、ユーザに選択した理由を知らせることができると考えられる。今後こういったことを検討する予定である。現在のシステムではプログラムの可読性を高めるターゲットとして、実際に計算に用いられる汎用言語ではなく、中間言語である。従って中間言語のレベルでは可読性の高いものが得られても変換後のプログラムはその可読性がどれくらい保たれるかについても更に検討すべきであると考える。

参考文献

[1] 佐川暢俊、金野千里、梅谷征雄: 数値シミュレーション言語DEQSOL、情報処理学会論文誌、第30巻第1号別刷、Vol.30, No.1, pp.36-45(1989).

[2] 大河内俊夫、金野千里、猪貝光祥: 数値シミュレーション向き高水準言語DEQSOLの分散メモリ型並列計算機向けトランスレーションに関する一考察、並列処理シンポジウムJSPP'93、pp.39-46(1993).

[3] 金野千里、梅谷征雄、太田忠、深田肇、山賀晋、池田美以子: 対話型数値シミュレーションシステム：ビジュアルDEQSOL、情報処理学会誌第33巻、第7号別刷、pp929-942(1992).

[4] Chisato Konno, Yukio Umetani, Mitsuyoshi Igai and Tadashi Ohta: INTERACTIVE/VISUAL DEQSOL: INTERACTIVE CREATION, DEBUGGING, DIAGNOSIS AND VISUALIZATION OF NUMERICAL SIMULATION, Mathematics and Computers in Simulation 31, pp.353-369(1989).

[5] Rice, J.: Solving Elliptic Problem Using ELLPACK, Springer-Verlag, p.497(1984).

[6] Cook, G.O.: ALPAL: A Tool for the Development for Large-Scale Simulation Codes, UCID-21482, Lawrence Livermore National Laboratory (1988).

[7] P.Baras, J. Blum, J.C.Paumier, P. Witomski: EVE:An Object-Centered Knowledge-Based PDE Solver, IMACS (1992).

[8] S.Kawata, K.Iijima, C.Boomnee, Y.Manabe: Computer-assisted scientific-computation/simulation-software-development system - including a visualization system -, IFIP Transactions Vol.A-48, pp.145-153(1994).

[9] 川田重夫、飯島邦彦、Choompol Boonmee、真鍋保彦： 記号処理手法による数値シミュレーションコード開発支援システム、情報処理学会第34回プログラミングシンポジム, pp.61-71(1993.1).