ハイパフォーマンス
コンピューティング 54－3
（１９９４．１２．１４）

数値プログラムの解析により データ形式を得て行う可視化

真鍋保彦，Choompol Boonmee，川田重夫
長岡技術科学大学 工学部 電気系

数値計算プログラムの計算結果のグラフ化を行うことは，結果の考察等を行う上で欠くことのできない重要なことである．数値データの出力形式等を変更する場合，グラフ化プログラムの一部を変更したり，出力データの加工等を行う必要が生じる．また，シミュレーション問題ごとにグラフ化プログラムを用意することも，グラフ化プログラム自身が，多くの場合，同一内容であることから非効率的である．本研究では，数値データの形式に関する情報が数値計算プログラムの中に含まれていることに着目し，数値計算プログラム自体を解析しデータ形式の情報を抽出する．この情報を利用し，変更されたデータ形式に自動的に追従するようなグラフ化を行い，グラフ化の効率の向上を図る．

Analysis of Computing Program for Numerical-Data-Format Acquisition in Scientific Visualization

Yasuhiko Manabe, Choompol Boonmee and Shigeo Kawata
Department of Elecrical Engineering,
Nagaoka University of Technology,
1603-1 Kamitomioka, Nagaoka, Niigata. 940-21, Japan.
E-mail: kawata@voscc.nagaokaut.ac.jp

Graphics of numerical data are essential in numerical analyses. If we change the numerical-data format and so on, we need to modify a part of a numerical program or the numerical data. On the other hand it is inefficient to prepare a visualization program for every simulation problem. In our study, we pay an attention to a fact that the computing program contains the data format information. We extract the information from it, and we use the information to perform the visualization. Our aim is to improve the visualization efficiency. In this paper we propose this idea and present a visualization system based on the idea.

1. 導入

計算機の高性能化，低価格化に伴い数値シミュレーションが様々な分野にわたって行われるようになってきた．数値シミュレーション結果の考察および検討を行うにあたってグラフ化を行うということは重要である．数値シミュレーションを行うためにプログラムを作成するが，同時にグラフ化を行うためのプログラム，いわゆるグラフ化プログラム[1,2]も用意する必要がある．このグラフ化プログラムは，計算プログラムの出力するデータの形式に合わせて作成する．だが，様々な検討を行うために，出力データの数の増減を行い，計算プログラムの数値データの出力ルーチンの一部を変更すると，データの出力形式が変化する．このような場合，グラフ化プログラムの一部をそれに合わせて変更する必要が生

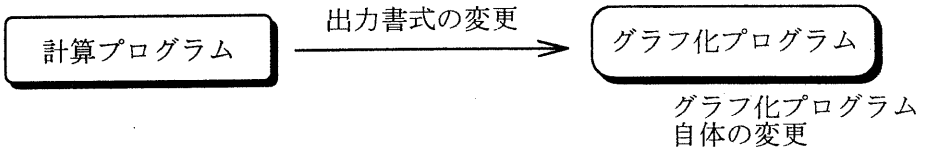

図1. 出力形式の変更に伴うグラフ化プログラムの変更

じる．これはシミュレーション結果の考察および検討を行う上で手間と時間がかかる（図1）．

このような手間を軽減するためには，グラフ化プログラムの方で数値データの形式の変更に自動的に対応するような方式を取る方法が考えられる．そのためにはグラフ化を行うに先立って数値データの形式を得る必要がある．数値データは計算プログラムを実行した結果として得られる．そのプログラム中のデータ出力ルーチンには出力データの出力書式および出力回数等の情報が含まれているはずである（図2）．

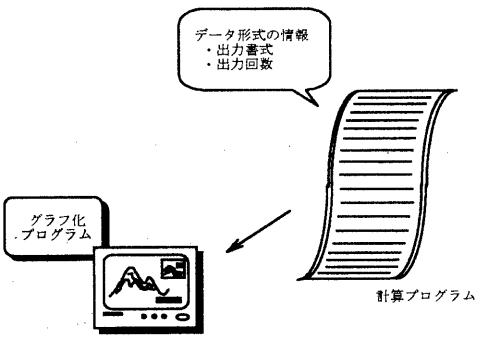

図2. 計算プログラムに含まれるデータ形式

本研究では，この点に着目し，計算プログラム自体を解析することにより，数値データの出力形式を得ることを提案し[3-5]，この提案をもとにして作成したシステムについて紹介する．

ここで，プログラム解析を行う理由について述べる．

1) 他人の作ったプログラムがあるが，データの出力形式がはっきりしない場合．

他人が作った計算プログラムが存在するが，そのプログラムが出力する数値データの形式がわからないことがある．データ形式についての適切なドキュメントがあれば別であるが，そうでなければ，そのプログラムを利用するユーザがプログラムを調べて

データの出力形式を特定しなければならない．しかし，大規模なシミュレーションプログラムとなると，そのプログラムリストは数千行や数万行のオーダとなる．よって，リストを調べてデータの出力形式を特定するのは，非常に困難であり時間もかかる．このような処理は高速なコンピュータに任せることができる．プログラムに関するドキュメントがあっても，その中からデータ形式を探す手間を省略することができる．

2) 自作のシミュレーションプログラムが存在するが，データの出力形式の変更を頻繁に行う場合．

シミュレーションを行う上では，様々な検討をおこなうために，多くのデータを計算し出力する．検討をおこなう上で，新たに必要となったデータを出力することも，不必要となったデータを出力しないようにすることもある．このような場合，数値データの出力情報が頻繁に変動することになる．それに合わせてグラフ化用のプログラムを書き換えるなどの作業が増えてしまう．そこで，グラフ化プログラムの方でこのような出力形式の変動に自動的に対応するようにすればグラフ化の効率が向上する．

3) 古い過去の自作のプログラムがあるが，データ形式の特定に時間がかかる場合．

以前にユーザ自身が作成したプログラムがあるが，プログラムリストの中からデータ形式を特定化するのに時間を要する．

以上のことからプログラムを解析して保存データ形式を抽出することは，数値データのグラフ化の効率化につながると考えられる．

2. グラフ化に必要な情報

本稿では，"データ形式"という言葉は

1) 出力されている数値データの型（整数型，浮動小数点型等）

2) グラフ化に必要なデータの保存順番，変数名

を含んだ，グラフ化データを読み込むのに必要な情報であると定義する．

数値データのグラフ化に必要な情報は 1) 数値データを読み込むために必要な情報，と，2) グラフそのものを描くために必要な情報 とに大きく分けられる．これらについて以下で考える．

2.1 数値データの読み込みに必要な情報

1) 数値データファイル名

数値データをどのファイルから読むのかを特定するために，読もうとする数値データファイル名が必要である．

2) 数値データの書式

数値データを適切に読むために，出力されたデータがどのような型で出力されているのかを知る必要がある．

3) 数値データを読む順番と回数

数値データをどういった順序で，何回読み込むのかを知る必要がある．

上に述べた1)～3)の情報はプログラムを解析することにより得ることができる．1)については，計算プログラム中で数値データファイルをオープンしている文を抽出し解析すれば得られる．2)については，データを出力する際に使われている書式文の書式指定子を抽出することにより知ることができる．3)については，データを出力している文が繰り返し文の制御範囲内に含まれていれば，その繰り返し文の繰り返し回数を解析することで知ることができる．

1)において，ファイル名が指定されている部分がファイル名そのものでなく，変数であるような場合，3)において繰り返し回数が変数で記述されていた場合は，そのままでは変数の値を特定することはできない．この場合，変数の値が設定されている部分を解析する必要がある．さらに，変数に設定する値をファイルから入力している場合もある．この場合には，必要な値をファイルから入力するようにしなければならない．

しかし，このようにしてすぐに必要な情報が得られるとは限らない．1)や3)において，変数の値がユーザからの入力によって決定される場合には，解析により値を決定することは不可能である．この場合には，必要な値をユーザに直接指定してもらわなければならない．

2.2 グラフ化に必要な情報

グラフ化に必要な情報について述べる．

1) 座標値および物理量

2) 物理量がスカラ量であるかベクトル量であるかという情報

3) グラフ化方法

グラフ化に際し，どういった描画方法を使用するのかを知る必要がある．

4) 数値データとグラフ化プログラム内部のグラフ化用配列との割り付けの情報

グラフ化を行うために，グラフ化プログラムにグラフ化用配列を用意し，これに座標，物理量を格納する．こうすることによりグラフ化プログラムを変更せずに，様々なデータ形式に対応できることになる．この概念図を図3に示す．読んでいるデータが，このうちのどこに格納されるべきなのかを知る必要がある．このための情報を知る必要がある．

上に述べた情報のうち，1)は，数値データより直接得ることができる．2)から4)は，ユーザが指定しなければ，その情報を得ることはできない．

3. プログラム解析

3.1 プログラム解析により得られる情報

プログラム解析により得られる情報について述べる．本研究ではFortran言語を解析の対象としているので，それに対して述べる．

1. ループ構造

DO文と対応する端末文(CONTINUE文)との関係からループ構造がわかる．

2. データ出力の繰り返し回数

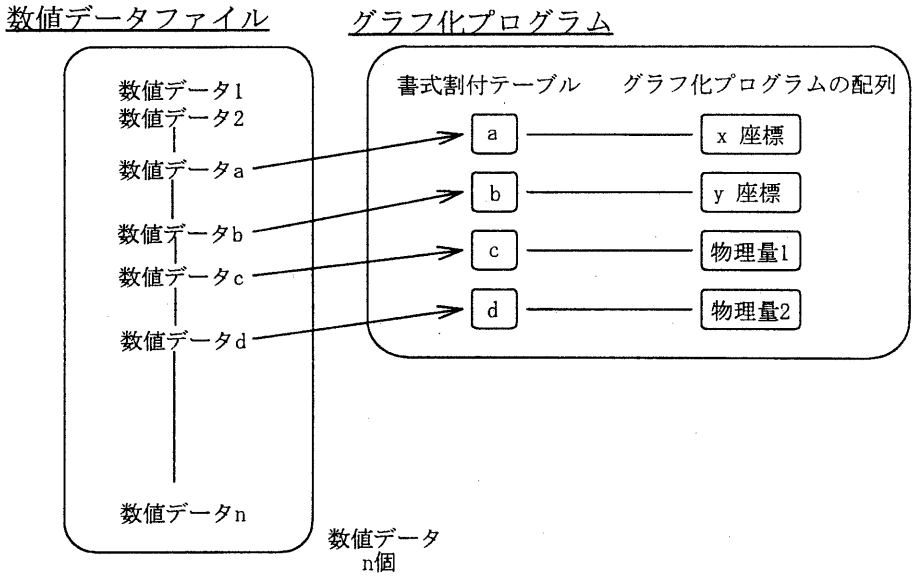

図3. グラフ化用配列と書式の割り付けの様子

DO文の初期値，終値および増分が数値で直接記述されていれば，すぐに繰り返し回数がわかる．変数で記述されている場合には，それらの値を確定するために更に解析をおこなわなければならない．

a) PARAMETER文

定数を定義するためにPARAMETER文がある．値が不明の変数がここに存在し，途中でその値が変更されていないのであれば，PARAMETER文を解析することにより値を確定できる．

b) COMMON文

値が不明の変数が，COMMON文の指定の中に含まれていれば，それに対応する呼び出し側のCOMMON文の中に含まれる変数が設定されている部分を検索することにより，値を特定できる．例えば，呼び出し元でCOMMON /A/X,Y,Zとなっており，呼び出された出力ルーチン側でCOMMON /A/A,B,Cとなっているとする．このとき，出力ルーチン中の変数Bの値が確定しない場合，呼び出し側のYという変数が設定されている部分を探し，値を確定する．

c) サブルーチン呼び出しの引き数

値が特定していない変数が直接サブルーチン呼び出しの際の引数として渡されていることがある．例えば，呼び出し側でCALL OUTOUT(A,B,C)と記述されており，サブルーチン側でSUBROUTINE OUTPUT(X,Y,Z)と記述されていたとする．ここでサブルーチンOUTPUT内でZの値が確定していないときは，呼び出し側のCを確定できればよいことになる．

d) 未確定変数のファイルからの入力

変数の値の設定は，プログラム中で行われるだけとは限らない．ときにはファイルから具体的な値を読む場合もある．これについては，ファイルをオープンしているOPEN文とデータを入力するREAD文を解析することによりファイルから読んでいるということがわかる．

e) 実行時においてユーザによって変数に設定されるべき値を入力

未確定の変数の値が，プログラム中で設定することもなく，ファイルから読むこともなく，実行時にキーボードから具体的な値を入力する場合もある．このような場合には，プログラムを解析することにより値を確定することは不可能であるので，ユーザが直接指定する必要がある．一般的には，この場合は非常に少ないと考えられる．

3. 出力しているデータの変数名

WRITE文に出力する変数が記述されている．それを取り出すことができる．

4. データを出力している書式

Fortranでは出力書式を定めるFORMAT文がある．それと対応するWRITE文とから出力書式がわかる．

5. 複数の出力ルーチン

計算プログラムにおいては，複数の出力ルーチンが使われる場合がある．例えば，ある出力ルーチンAとBがあり，ある条件のもとではAを使用し，それ以外ではBを使用するといったものである．

1) 複数の出力ルーチンがプログラム中に順次現れる場合

例えば，ある計算プログラムがA，B，Cという3つの出力ルーチンを持っていたとする．これらのルーチンがプログラム中で特別な条件などなく順次現れて使用されている場合には，A，B，Cそれぞれのルーチンを解析することで，出力形式の情報を得ることができる．

2) 複数の出力ルーチンがそれぞれある条件のもとで使用される場合

例えば，変数の値により出力ルーチンを選択するような場合がこれにあてはまる．この場合，変数の値を特定する必要がある．その値がプログラムを実行することにより決定される"動的"な場合と，単なる代入文や算術式のみで記述され，プログラム実行により値が変動することのない"静的"な場合とがある．"静的"なものに関しては，解析によりその変数の値を得ることが可能だが，"動的"なものに関しては不可能である．

4. システム概要

ここで，システムの概要について述べる．計算プログラムを解析することにより，数値データの出力構造を自動的に抽出する．このことにより，計算プログラムの変更によって出力データ形式が変化することに起因するグラフ化プログラムの変更の手間を軽減する．この考えにもとづき，本研究ではグラフ化効率の向上を図ることを目的としている．

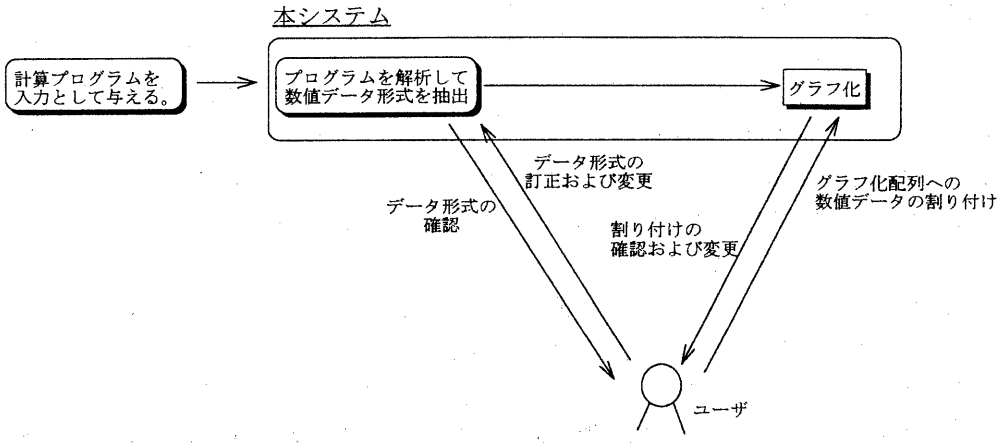

図4. グラフ化に至るまでの流れ

図4にプログラム解析からグラフ化に至るまでの流れを示す．システムに計算プログラムを入力として与える．本システムはそのプログラムを解析し，データを出力していると思われる部分を取り出す．次にその部分を解析し，データの出力書式，変数および出力回数等の情報を抽出し，その結果をファイルに出力する．このファイルを我々は書式情報ファイルと呼ぶ．この内容はユーザに提示される．システムがこの解析により得たデータ形式に正しくない箇所はないかどうかを，ユーザはチェックすることができる．正しくない箇所があれば，ユーザがエディタ内で編集し，その内容を新たに書式情報ファイルの内容として使用できるようになっている．次に，図3に示した概念に基づき，数値データをグラフ化用配列に割り付ける．システムは書式情報ファイルの内容をもとにしながら，数値データファイルより数値データを読む．

書式情報ファイルの一例を図5に示す．VARで始まる行は数値データファイルよりデータを読み込むための制御文である．この文はコロン（：）で区切られ，5つのフィールドからなっている．1番目のフィールドはデータファイルからデータを読むというVAR文そのものの宣言である．VARの後には複数のVAR文を識別するための番号が続く．2番目のフィールドはデータを入力するための書式を表す．この書式文字列はC言語のscanf()関数の書式指定子に準じている．たとえば，'%d'であれば，整数型でデータを読み，'%f'であれば浮動小数点型で読むことになる．3番目の

```
VAR1:%d:(I):OUTPUT :v_data.70x70
VAR2:%d:(J):OUTPUT :v_data.70x70
@
        LOOP1:70
                LOOP2:70
                VAR3:%e:((I-1)*DX):OUTPUT :v_data.70x70
                VAR4:%e:((J-1)*DX):OUTPUT :v_data.70x70
                VAR5:%e:(P[I][J]):OUTPUT :v_data.70x70
                VAR6:%e:(U[I][J]):OUTPUT :v_data.70x70
ENDLOOP2
ENDLOOP1
```

図5. 書式情報ファイルの一例

フィールドは，数値計算プログラムの中でデータを出力するために使われている変数名を表す．4番目のフィールドは，数値計算プログラムの中でデータが出力されているルーチンの名前を表す．5番目のフィールドは，データファイル名を表す．以上のことから，図5.の一行目に関して解釈すると次のようになる．「第1番目の数値データを読むための宣言であり，その書式は'%d'（整数型）であり，それはプログラム中のOUTPUTというルーチンの中にIという変数名で出力されたものである．そのデータを読むためにはv_data.70x70というファイルを使用せよ．」

LOOPで始まる行はデータ読み込みの繰り返しを記述するために用いる．LOOPの後にはループ構造の識別のための通し番号が付く．コロンで区切られた2番目のフィールドは繰り返すための回数を表す．LOOPの端末文としてENDLOOPを用いる．LOOPは対応するENDLOOPが現れるまでの内容を繰り返し実行する． 図4. の場合は，LOOP1～ENDLOOP1の中にさらにLOOP2～ENDLOOP2が現れており，二重ループになっている．

5. まとめ

計算プログラムを解析して数値データの出力形式を抽出し，グラフ化の際に利用する手法を提案し，それについて述べた．計算プログラムを解析して得られる情報からグラフ化を行うことは可能であり，このことがグラフ化の効率の向上につながると考えられる．現在は解析の対象としてFortran言語を採用しているが，C言語等への拡張は可能である．

参考文献

[1] Advanced Visual System Inc.: AVS User's Guide, Waltham (1992).

[2] 日立製作所：マニュアル　数値シミュレーション用グラフ作成プログラム SGRAF E2 サブルーチン編，日立製作所 (1992).

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

[4] 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ウム，pp.61-71(1993).

[5] Choompol Boonmee，真鍋保彦，渋井俊昭，釣谷浩之，森正孝，川田重夫：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－，情報処理学会研究報告ハイパフォーマンスコンピューティング No.53，pp.1-8(1994).