FlexPDE version 5

コマンドリファレンス

開発元：PDE Solutions Inc.
販売代理店：（株）ライトストーン

本書は *Scientific WorkPlace* を用いて作成されています。

目次

第 1 章

序文

第 II 部では FlexPDE の problem descriptor（スクリプト）を構成する各種要素について詳細を記述します。これら要素の使用法につきチュートリアル的な説明はここでは行いません。操作法に関する情報については操作ガイドマニュアルを、用例に基づくチュートリアル情報についてはユーザガイドマニュアルをご参照ください。

第 2 章

はじめに

FlexPDE はスクリプトベースのシステムであり、“problem descriptor”、あるいは“**スクリプト**”と呼ばれるテキストファイルから方程式、ドメイン、付帯定義情報、グラフィックス出力仕様に関する記述を読み込みます。

Problem descriptor ファイルは通常 FlexPDE のエディタを用いて作成しますが、一般的な ASCII テキストエディタで作成しても構いません。その他のワープロソフトについては、フォーマット用のコードを含まない純粋なテキスト形式での出力が得られる場合に限って使用することができます。

Problem descriptor は当初 Robert G. Nelson によって開発された**自然言語**に基づいています。この言語については Dr. Gunnar Backstrom の書籍“Field of Physics by Finite Element Analysis - An Introduction”中にも解説があります。またこの言語は Lawrence Livermore National Lab の PDS2、及び PDEase2 システムにおいて使用されてきました。

FlexPDE の進化に伴い多くの拡張がこの言語に対し施されてきました。本コマンドリファレンスでは現行 FlexPDE で実装されている言語仕様について説明します。

Problem descriptor の記述に用いられる自然言語はある意味コンピュータプログラム言語に似ていますが、はるかに単純なものです。偏微分方程式の初等コースを学んだことのある人であれば、学生であれエンジニアであれ研究者であれ、1 時間もあれば簡単な problem descriptor を準備でき、問題の解法を進めることができるようになります。

FlexPDE の problem descriptor は**有限要素モデル**（Finite Element models）を生成するための簡易言語でもあります。Descriptor 上のステートメントに基づき FlexPDE は問題を解くための数値解法プロセスを生成します。

FlexPDE problem descriptor の言語はプロシジャ型言語ではなくリレーショナル型の言語である点に注意してください。ユーザは種々のシステム要素が相互にどう関連しているかを記述します。

解を導くためにどういうステップを取るべきかのシーケンス（プロシジャ）を記述するわけではありません。Problem を構成する要素間の関係に基づき、FlexPDE が解を求めるために必要なステップのシーケンスを決定します。

FlexPDE は problem descriptor 上の要素に対し種々の仮定を置きます。

例えばある変数[*1]名が `VARIABLES` セクション中に現れたとすると次が仮定されます。

- この変数はスカラー場を表し、problem のドメイン上で値が定義される。
- 計算メッシュを構成する節点（nodes）の間は有限要素補間によって近似が可能である。
- ドメイン上で変数の値は連続である。
- 変数の振舞いを規定する偏微分方程式が与えられる。

またある定義文が `DEFINITIONS` セクションに現れた場合、次が仮定されます。

- その名標は偏微分方程式の系に付帯する。
- ドメイン上で不連続な場合がある。
- 偏微分方程式に従うとは限らない。

以下のセクションでは problem descriptor を構成する際の各種ルールについて記述します。

2.1 Descriptor ファイルの作成

FlexPDE 用の problem descriptor ファイルを作成/編集する上で最も簡便なのは FlexPDE 内蔵のエディタを使用する方法です。

新たな descriptor ファイルからはじめる場合は FlexPDE のメインメニューバーより“File | New Script”と操作してください。

一方、既存の descriptor を編集するアプローチを取るなら“File | Open Script”と操作します。

新たな descriptor を用意する場合に最も楽な方法は、類似の問題に対する既存の descriptor をコピーした上で、それに対し必要な変更を加えて行くアプローチです。

FlexPDE 内蔵の**エディタ**は Windows のメモ帳エディタに近いもので、フォーマット用の文字コードを全く含まない純粋な ASCII テキストファイルを生成します。Descriptor ファイルは任意の ASCII テキストエディタを用いて、あるいは純粋な ASCII テキストファイルをエクスポートでき

[*1] 訳者注： 原文では variable と書かれていますが、以降、一般に“変数”と書かれている場合は“従属変数”と解釈してください。

る任意のワープロソフトを用いて準備することもできます。なお、フォーマット用の文字コードが埋め込まれていた場合には、FlexPDE の構文解析中にエラーが生ずることもあるので注意してください。

FlexPDE 内蔵のエディタはスクリプトを見やすいものとするため構文に応じた強調標示を行います。識別されたキーワードは赤で、コメントは緑で、文字列は青で標示されます。

2.2 ファイル名称と拡張子

Problem descriptor ファイルの名称には ASCII コードを使用し、日本語用のコードは使用しないでください。

名称中に空白を入れるのは OS によっては許されているものもありますが、トラブルの元となるので空白は使用しないでください。

問題の内容がわかるような名称を選択するのがベストと言えます。

Problem descriptor ファイルは拡張子“.pde”を持ったものでなくてはなりません。

FlexPDE の内蔵エディタを使用してファイルを保存した場合には、拡張子“.pde”は自動的に付加されます。

他のエディタを使用する場合には拡張子として“.pde”を明示する必要があるので注意してください。

2.3 Problem descriptor の構成

Problem descriptor は problem を関連する項目ごとに**セクション**に分割した形で構成します。

それぞれのセクションは名称から始まり、それに続く形で 1 つ、または複数のステートメントが配置されます。

許容されるセクション名は次の通りです。

TITLE	Problem の表題を規定する。
SELECT	各種のオプションや制御項目（controls）を設定する。
COORDINATES	座標系を規定する。
VARIABLES	Problem における変数名を規定する。
DEFINITIONS	付帯する量やパラメータ値を規定する。
INITIAL VALUES	変数の初期値を設定する。
EQUATIONS	偏微分方程式の系を定義する。
CONSTRAINTS	積分に対する制約条件を規定する（オプショナル）。
EXTRUSION	ドメインを 3 次元に押出す（extrude）。
BOUNDARIES	2 次元、または 3 次元ドメインの境界を規定する。
RESOLVE	メッシュ生成に関する制御を補佐する（オプショナル）。
FRONT	メッシュ生成に関する制御を補佐する（フロント用、オプショナル）。
TIME	タイムドメインの規定。
MONITORS	途中経過のグラフィックス表示を規定する。
PLOTS	最終的なグラフィックス出力を規定する。
HISTORIES	ヒストリグラフィックスを規定する。
END	Descriptor の末尾であることを示す。

特定の problem descriptor 中で使用されるセクションの数は問題ごとに異なります。しかし Boundaries セクションと End セクションだけはすべてのファイル中に存在していなくてはなりません。

これらセクションの並べ方には多少の柔軟性はありますが、上の表に記した順に従っていただくのが間違いのない方法です。

DEFINITIONS と SELECT は複数回現れても構いません。

Descriptor は上から下に処理されるため、前方向への参照を含めることはできません。定義中で変数名やその他の名称を参照する場合、それらは先行するセクション中で、あるいは同一セクション内の先行部で定義済みでなくてはなりません。

2.4 Problem descriptor の様式

これは必須というわけではありませんが、次のような字下げ（indentation）を用いることを推奨します。

```
section 1
  statement
section 2
  statement 1
  statement 2
      •
      •
section 3
  statement 1
  statement 2
      •
      •
```

この様式に従うことによって、descriptor ファイルは作成者にとっても第 3 者にとっても見やすくわかり易いものとなります。

2.5 大文字/小文字の区別

引用符で囲まれた文字列を例外とするなら、problem descriptor 中で用いられる単語や文字列において大文字/小文字の区別はありません（引用符で囲まれた文字列の場合には problem descriptor 中での表記通りに出力されます）。

大文字と小文字は同一視されます。例えば `variables`, `VARIABLES`, `Variables`, `VaRiAbles` の間に区別はありません。

2.6 Include ファイル

FlexPDE は外部のファイルを読み込むために problem descriptor 上で INCLUDE 文を使用することを許しています。

```
#INCLUDE "filename"
```

というステートメントがあった場合、この文は該当ファイルの中身によってそっくり置き換えられます。

該当ファイルが descriptor とは別のフォルダ中に置かれている場合には、そのファイルに対するフルパス名が指定されなくてはなりません。

include 文は descriptor 中のどこにあっても構いませんが、readability という視点からするなら、それは独立した行に記述されるべきです。

いくつかの descriptor で共通の定義情報を使用するような場合にこの機能は有効と言えます。

Note: FlexPDE には大文字と小文字の区別はありませんが、OS によってはそれらを区別する可能性があります。外部ファイル名の指定に際しては OS 側の慣行に従ってください。

2.7 簡単な用例

FlexPDE descriptor ファイルの感触をつかんでいただくため、ここでは正方形領域上での熱流モデルを作成してみます。

熱流方程式は

```
div(K*grad(Temp)) + Source = 0
```

で与えられます。この場合の温度に関する方程式としては

```
Temp = Const - x^2 - y^2
```

が使用できます。ここに K は定数、Source = 4*K とします。

ここでは熱伝導率 K の値を 1、熱源としては一様に分布する単位面積当り 4 単位の熱源を想定し、正方形領域を定義します。

一方、境界値としては次を設定します。

```
Temp = 1 - x^2 - y^2
```

この例の場合、解析解が知られているので、FlexPDE の解の精度を検証することができます。

Descriptor の記述は次のようになります。

```
{ ********************************************************************
  SIMPLE.PDE
  This sample demonstrates the simplest application of FlexPDE
  to heatflow problems.
******************************************************************** }

TITLE "Simple Heatflow"

VARIABLES
  temp                { Identify "Temp" as the system variable }

DEFINITIONS
  k = 1               { declare and define the conductivity }
  source = 4          { declare and define the source }
  texact = 1-x^2-y^2  { exact solution for reference }

INITIAL VALUES
  temp = 0            { unimportant in linear steady-state problems,
                      but necessary for time-dependent or nonlinear
                      systems }

                      { define the heatflow equation :}
EQUATIONS
  div(k*grad(temp)) + source = 0
```

```
                    { define the problem domain:  }
BOUNDARIES
  REGION 1          { ...  only one region }
    { specify Dirichlet boundary at exact solution:  }
    VALUE(temp)=texact
    START(-1,-1)    { specify the starting point }
    LINE TO (1,-1)  { walk the boundary }
      TO (1,1)
      TO (-1,1)
      TO CLOSE      { bring boundary back to starting point }
MONITORS
  CONTOUR(temp)     { show the Temperature during solution }
PLOTS               { write these plots to disk at completion:  }
  CONTOUR(temp)     { show the solution }
  SURFACE(temp)     { show a surface plot as well }
                    { display the solution error :}
  CONTOUR(temp-texact) AS "Error"
                    { show a vector flow plot:  }
  VECTOR(-dx(temp),-dy(temp)) AS "Heat Flow"
END                 { end of descriptor file }
```

第 3 章

Descriptor の要素

FleXPDE に対し問題の特性を記述する problem descriptor（“**スクリプト**”）は、名称、記号、予約語、数値定数など、数多くの基本的な**要素**から構成されています。以下のセクションではこれらの要素について説明します。

3.1 コメント

Problem descriptor 中にはコメント文を挿入することができます。

複数行からなるコメントはファイルの任意の位置に配置することができます。コメントはその両端を中括弧 { }、あるいはペア記号/*と*/で囲むことによって構成されます。コメントのネスティングも可能ですが、{ で始まったコメントは } で、/*で始まったコメントは*/で終わらせる必要があります。

Example:

```
{ this is a comment
so is this.
}
```

単行のコメントは感嘆符!を用いて設定することができます。この場合、!から行末までがコメントとみなされます。行の先頭に!を置いた場合には行全体がコメント行となります。これは C++ における“//”と同列の機能です。

Example:

```
!  this is a comment
this is not
```

Problem descriptor 中の一部の行をコメント化することで機能の一部を一時的に無効にすることができます。スクリプトの開発、デバッグに有効な場合があります。

3.2 予約語と記号

FlexPDE には特別な意味を持った予約語や記号が多数存在します。コメントや純粋な文字列（ストリング）として利用する場合を除き、これらの予約語はその目的に合致した形で使用されなくてはなりません。

ABS	ALIAS	AND	ANGLE
ARC	ARCCOS	ARCSIN	ARCTAN
AS	AT	ATAN2	
BESSJ	BESSY	BINTEGRAL	BOUNDARIES
BY			
CDF	CENTER	CLOSE	CONIC
CONSTRAINTS	CONTOUR	COORDINATES	COS
COSH	CROSS	CURL	
DEBUG	DEFINITIONS	DEGREES	DEL2
DELTAT	DIFF	DIR	DIRECTION
DIV	DNORMAL	DOT	DTANGENTIAL
ELEVATION	ELSE	END	ENDTIME
EQUATIONS	ERF	ERFC	EXCLUDE
EXP	EXPINT	EXPORT	EXTRUSION
FEATURE	FILE	FINISH	FIT
FIXED	FOR	FROM	
GAMMAF	GLOBALMAX	GLOBALMIN	GRAD
GRID			
HISTORIES	HISTORY		
IF	INITIAL	INTEGRAL	INTEGRATE
INTSTRING			
JACOBIAN	JUMP		
LAMBDA	LAYER	LAYERED	LIMITED
LINE	LIST	LN	LOAD
LOG10			
MAGNITUDE	MAX	MESH_DENSITY	MESH_SPACING
MIN	MOD	MONITORS	MOVE
NATURAL	NEUMANN	NORMAL	NOT
OFF	ON	OR	

PERIODIC	PI	PLOTS	POINT
PRINT	PRINTONLY		
RADIANS	RADIUS	RAMP	REGION
REPEAT	REPORT	RESOLVE	
SCALAR	SELECT	SIGN	SIMPLEX
SIN	SINH	SPLINE	SPLINETABLE
SPLINETABLEDEF	SQRT	STAGE	STAGED
START	SUM	SUMMARY	SURFACE
SWAGE			
TABLE	TAN	TANGENTIAL	TANH
TECPLOT	THEN	TIME	TITLE
TO	THRESHOLD	TRANSFER	TRANSFERMESH
UNORMAL	UPULSE	URAMP	USTEP
VAL	VALUE	VALUES	VARIABLES
VECTOR	VELOCITY	VERSUS	VIEWANGLE
VIEWPOINT	VOID	VTK	VTKLIN
XCOMP			
YCOMP			
ZCOMP	ZOOM		

3.3 セパレータ

White space

スペース、タブ、空白行はしばしば“white space”と呼ばれますが、これらはセパレータとして扱われ、readability 向上のため problem descriptor 中で自由に使用することができます。White space が複数連続した場合、FlexPDE はそれらを単一の white space として処理します。

コンマ

コンマはリスト中の要素を区切るために用いられます。コンマは descriptor の構文上、明示された場所でのみ使用してください。

セミコロン

セミコロンはラベルやステートメントの末尾が明確でない場合に、それを明示する上で使用されます。数学的な量を表す 2 つの名標が演算子をはさまない形で white space によって区切られていた場合、FlexPDE は一つの等式が終わり別の等式が開始されるものと解釈します。一方、それらの名

標の間に演算子が置かれていた場合、たとえそれらが別個の行にあったとしても、FlexPDE は一つの等式がまだ継続していると解釈することになります。もし新たな等式が“–”のような演算子で始まり、それが別の等式に続く形になっているのであれば、最初の等式の末尾にはセミコロンを配置する必要があります。

3.4 文字ストリング

文字ストリングはオプショナルなユーザ定義ラベル（ハードコピー/ソフトコピー双方に出力される）を設定する際に用いられます。

指定された文字ストリングから生成されるラベルは大文字/小文字の別も含め、入力された通りに出力されます。

文字ストリングはラベル情報をシングルクォーテーション（’ ’）、あるいはダブルクォーテーション（" "）で囲んだものです。ダブルクォーテーションで始まる文字ストリングはダブルクォーテーションで終わるものでなくてはならず、シングルクォーテーションで始まる文字ストリングはシングルクォーテーションで終わるものでなくてはなりません。

文字ストリングは英数字、セパレータ、予約語、記号の任意の組合せで構成できます。ここで言う記号の中にはクォーテーションマークも含みますが、それを含める場合には次の条件が付きます。すなわちダブルクォーテーションで始まる文字ストリング中にはシングルクォーテーションマークを含めることができ、一方、シングルクォーテーションで始まる文字ストリング中にはダブルクォーテーションマークを含めることができます。

Example:

```
TITLE "This is a literal 'string' used as a problem title"
```

3.5 数値定数

整数

整数は xxxxxx の形式でなくてはなりません。ここに x は 0 から 9 までの 10 進数を意味します。整数の定数には最大 9 個の数字を与えることができます。

小数

小数は xxxxx.xxx の形式でなくてはなりません。ここに x は 0 から 9 までの 10 進数を、“.”は小数点を意味します。小数にはコンマを含めることはできません。欧州の慣行に従ってコンマを小数点として使用するとエラーが発生します。コンマの使用は項目を区切る目的にのみ制限されています。小数全体としては 308 桁の数字まで持つことができますが、小数点より左側に位置する数字の

数は 0 から 9 までに限られます。FlexPDE は先頭から 15 桁の数字のみを有効なものとして使用します。

浮動小数

工学記法（engineering notation）に基づく浮動小数は xxxxxEsyyy という形式でなくてはなりません。ここに x は 0 から 9 までの 10 進数、または小数点“.”を、y は 0 から 9 までの 10 進数を、E は指数セパレータ、そして s はオプショナルな符号記号を表します。浮動小数の表記の中にコンマは使用できません。欧州の慣行に従ってコンマを小数点として使用するとエラーが発生します。コンマの使用は項目を区切る目的にのみ制限されています。指数セパレータより左側の数は上記小数として扱われます。一方、指数セパレータより右側の数は 3 桁以下の整数として扱われます（小数点が含まれていてはいけません）。表現できる数は $1e$-307 から $1e308$ の範囲です。

3.6 組込み関数

関数と引数

関数を使用する場合には少なくとも 1 つの引数を指定する必要があります。引数としては数値定数、あるいは評価して数値となる数式が許されます。Problem descriptor 上では以下の関数を使用することができます。

3.6.1 解析関数

FlexPDE では次のような解析関数（analytic functions）がサポートされています。

関数	コメント
ABS(x)	絶対値
ARCCOS(x)	$\arccos(x)$　ラジアンを応答
ARCSIN(x)	$\arcsin(x)$　ラジアンを応答
ARCTAN(x)	$\arctan(x)$　ラジアンを応答
ATAN2(y,x)	$\arctan\left(\frac{y}{x}\right)$
BESSJ(order,x)	Bessel 関数 J
BESSY(order,x)	Bessel 関数 Y
COS(x)	$\cos(x)$　x の単位はラジアン *
COSH(x)	$\cosh(x)$
ERF(x)	誤差関数
ERFC(x)	余誤差関数
EXP(x)	指数関数

関数	コメント
EXPINT(x)	指数積分 $\mathrm{Ei}(x) \quad (x > 0)^{**}$
EXPINT(n,x)	指数積分 $\mathrm{En}(x) \quad (n \geq 0, x > 0)^{**}$
GAMMAF(x)	Gamma 関数 $\quad (x > 0)$
GAMMAF(a,x)	不完全 Gamma 関数 $\quad (a > 0, x > 0)$
LOG10(x)	$\log_{10}(x)$
LN(x)	自然対数 $\ln(x)$
SIN(x)	$\sin(x) \quad x$ の単位はラジアン *
SINH(x)	$\sinh(x)$
SQRT(x)	$\sqrt{x}$
TAN(x)	$\tan(x) \quad x$ の単位はラジアン *
TANH(x)	$\tanh(x)$

* 引数を度からラジアンに変換するには COS(x DEGREES) のように入力します。

** 定義については Abramowitz & Stegun, "Handbook of Mathematical Functions"を参照。

Examples:

Samples | Misc | Funtest.pde 参照

3.6.2 非解析関数

FlexPDE では次のような非解析関数（non-analytic functions）がサポートされています。

MAX(arg1, arg2)

この関数は 2 つの引数を必要とします。MAX の値は各点で評価され、2 つの引数のうち大きな方の値が関数値となります。

MIN(arg1, arg2)

この関数は 2 つの引数を必要とします。MIN の値は各点で評価され、2 つの引数のうち小さな方の値が関数値となります。

MOD(arg1, arg2)

この関数は 2 つの引数を必要とします。MOD の値は各点で評価され、$\dfrac{\arg 1}{\arg 2}$ の余りの値（modulo）が関数値となります。

GLOBALMAX(arg)

この関数は 1 つの引数を必要とします。Problem ドメイン全体における引数の最大値が関数の値となります。引数の成分が変化した場合には GLOBALMAX の値は再評価され表中に記憶されます。

GLOBALMAX_X(arg)

GLOBALMAX_Y(arg)

GLOBALMAX_Z(arg)

グローバルな最大値に対応する座標値が応答として返されます。探索の経過は表中に記憶されるので、GLOBALMAX、及びその座標値を求める関数コールを繰り返しても再計算は行われません。

GLOBALMIN(arg)

この関数は 1 つの引数を必要とします。Problem ドメイン全体における引数の最小値が関数の値となります。引数の成分が変化した場合には GLOBALMIN の値は再評価され表中に記憶されます。

GLOBALMIN_X(arg)

GLOBALMIN_Y(arg)

GLOBALMIN_Z(arg)

グローバルな最小値に対応する座標値が応答として返されます。探索の経過は表中に記憶されるので、GLOBALMIN、及びその座標値を求める関数コールを繰り返しても再計算は行われません。

RANDOM(arg)

この関数は 1 つの引数を必要とします。$(0, \arg)$ の間で一様に分布する擬似乱数値が応答となります。RANDOM 関数の利用が考えられるのは初期値の設定においてです。他のコンテクストにおいて使用した場合には収束に支障を来たす可能性があります。

SIGN(arg)

この関数は 1 つの引数を必要とします。引数の値が正であれば 1 が、負であれば -1 が関数値となります。

3.6.3 ユニット関数

FlexPDE では次のようなユニット関数がサポートされています。

USTEP(arg)

この関数（unit step function）は 1 つの引数を必要とします。USTEP の値は引数が正のときに 1、負のときに 0 となります。例えば USTEP(x-x0) は $x = x_0$ におけるステップ関数を表します。

UPULSE(arg1, arg2)

この関数（unit pulse function）は 2 つの引数を必要とします。UPULSE の値は arg 1 が正で arg 2 が負のときに 1、その他の場合は 0 となります。例えば UPULSE(t-t0, t-t1) は、$t_0 < t_1$ とするとき、t_0 から t_1 の範囲にわたるパルス関数を表します。

URAMP(arg1, arg2)

この関数（unit ramp function）は 2 つの引数を必要とします。URAMP は UPULSE と似たものですが、長方形ではなく傾斜を構成する点が異なります。

Examples:

Samples | Misc | Ufuntest.pde 参照

3.6.4 ストリング関数

FlexPDE は動的に文字列を構成するための最低限の機能を提供しています。

$integer

この関数は整数値 integer を表す文字列を応答します。この関数は連結（concatenation）演算子“+”と組み合わせる形で境界名やリージョン名を構成する際に有用です。次にその例を示します。

```
REPEAT i=1 to 4 do
  START "LOOP"+$i (x,y)
  <path_info>...
ENDREPEAT
```

これは以下と等価です。

```
START "LOOP1" (x,y) <path_info> ...
START "LOOP2" (x,y) <path_info> ...
START "LOOP3" (x,y) <path_info> ...
START "LOOP4" (x,y) <path_info> ...
```

Examples:

Samples | Misc | ArrayRepeat.pde 参照

3.6.5 FIT 関数

任意の引数に対し有限要素補間の計算を行う機能が 2 タイプ用意されています。

```
result = FIT(expression)
```

は現状の計算メッシュと基底関数（basis）を用い、与えられた数式 expression に対し有限要素フィットを計算します。個々のメッシュセル上で正しい積分値が得られるように節点の値を算出します。

```
result = FIT(expression,weight)
```

は基本的に FIT(expression) と同じですが、weight で示される平滑化のための拡散係数を伴います。weight の値としては 0.1 と 1.0 を試してみた上で適宜調整すると良いでしょう。

weight には空間座標変数や時間、その他の変数を含む任意の数式を指定できます。このようにすればメッシュの一部を選択的に平滑化できます。weight の値には明確な意味があります。それは変動を damping させる上での空間的な波長を意味します。weight の値よりもずっと小さな波長を持った空間的な変動は平滑化されるのに対し、それよりもずっと大きな波長の変動はほとんど変化しません。

Note: FIT() はドメイン全体にわたって連続なデータ表現を構成します。このためフィット対象のデータに不連続性があってもそれは保持されません。データに対して適切な材質パラメータを乗ずることによってフィットに適した連続的な関数を生成できる場合があります。

FIT() はノイズの多いデータを平滑化する上で有効です。これによってニュートン法による微分演算を安定化させることができます。また複雑な関数に伴う高価な再計算を回避することもできます。

節点の値を直接的に算出する SAVE 関数についても参照ください。

Examples:

Samples | Misc | fitweight.pde 参照

3.6.6 LUMP 関数

LUMP 関数は有限要素メッシュ上に引数の数式で規定される場を形成し、メッシュ上のそれぞれのセルに対し一つの値をセーブします。個々のセルに対して格納される値は引数の数式のそのセル上での平均値であり、該当セル内において定数として扱われます。

LUMP 関数はニュートン法における微分演算を安定化させる上で、あるいは複雑な関数に伴う高価な再計算を回避する上で有効です。

通常 LUMP 関数は DEFINITIONS セクション中で使用されます。

```
name = LUMP ( expression )
```

Examples:

Samples | Misc | lump.pde 参照

3.6.7 RAMP **関数**

RAMP 関数は URAMP 関数の変形であり、IF..THEN ステートメントに近い用法としたものです。

この関数は USTEP のような不連続関数や不連続な IF..THEN construct に代るものとして導入されました。

不連続な切替えは計算に際し種々の問題を引起す場合があります。特に時間依存型の場合に深刻な影響が及ぶため、その使用は極力避けるようにしてください。

FlexPDE は動的調整機能を持ったシステム（adaptive system）です。その論理は、タイムステップやセルサイズを小さくして行けば、いずれ解の振舞いは多項式で表現できるとの仮定に基づいています。しかし不連続性があるとこの仮定は成り立たなくなります。不連続なものはいくら解像度を上げていっても不連続です。瞬時のオン、オフは解の中に空間的/時間的高周波成分（現実システムの物理的限界をはるかに超えたものも含む）をもたらします。これによって計算が非常に遅くなるばかりか、物理的に意味のない結果が得られることもあります。

RAMP 関数を使うとある遷移幅である値を別の値へスムーズに変化させることができます。
IF..THEN 文に近いものですが、激しい不連続性は回避できます。

構文形式は次の通りです。

```
value = RAMP(expression, left_value, right_value, width)
```

これは次の表現

```
value = IF expression < 0 THEN left_value ELSE right_value
```

と等価ですが、唯一、遷移が width の幅をかけて線形に行われる点が異なります。

同様な機能を持ったものに SWAGE 関数がありますが、こちらの場合には導関数も含めてスムーズなものになります。

Examples:

Samples | Misc | Swagetest.pde 参照（RAMP の用例含む）

3.6.8 SAVE 関数

SAVE 関数は有限要素メッシュ上に引数の数式で規定される場を形成し、続いて行われる補間計算のために節点における値をセーブします。SAVE はドメイン全体にわたって連続なデータ表現を構成します。このため対象のデータに不連続性があってもそれは保持されません。

SAVE 関数はニュートン法における微分演算を安定化させる上で、あるいは複雑な関数に伴う高価な再計算を回避する上で有効です。

通常 SAVE 関数は DEFINITIONS セクション中で使用されます。

```
name = SAVE ( expression )
```

Note: SAVE() はドメイン全体にわたって連続なデータ表現を構成します。このためフィット対象のデータに不連続性があってもそれは保持されません。データに対して適切な材質パラメータを乗ずることによって保存に適した連続的な関数を生成できる場合があります。

積分値の整合性保持、及び変数に対する平滑化機能を持った FIT() 関数についても併せて参照ください。

3.6.9 SUM 関数

SUM 関数は数列の和を計算します。形式は次の通りです。

```
value = SUM( name, initial, final, expression )
```

引数 expression の値が評価され、name = initial,...,final の範囲で合計されます。

例えば

```
source = SUM(i,1,10,exp(-i))
```

というステートメントの場合、$\sum_{i=1}^{10} \exp(-i)$ が計算されます。

SUM 関数は配列データと共に使用することもできます。

```
DEFINITIONS
  A = ARRAY(1,2,3,4,5,6,7,8,9,10)
  source = SUM(i,1,10,A[i])
```

Examples:

Samples | Misc | Sum.pde 参照

3.6.10 SWAGE 関数

SWAGE 関数は USTEP のような不連続関数や不連続な IF..THEN construct に代るものとして導入されました。不連続な切替えは計算に際し種々の問題を引起す場合があります。特に時間依存型の場合に深刻な影響が及ぶため、その使用は極力避けるようにしてください。

FlexPDE は動的調整機能を持ったシステム（adaptive system）です。その論理は、タイムステップやセルサイズを小さくして行けば、いずれ解の振舞いは多項式で表現できるとの仮定に基づいています。しかし不連続性があるとこの仮定は成り立たなくなります。不連続なものはいくら解像度を上げていっても不連続です。瞬時のオン、オフは解の中に空間的/時間的高周波成分（現実システムの物理的限界をはるかに超えたものも含む）をもたらします。これによって計算が非常に遅くなるばかりか、物理的に意味のない結果が得られることもあります。

SWAGE 関数を使うとある遷移幅である値を別の値へスムーズに変化させることができます。またそれは導関数においても連続性を維持します。IF..THEN 文に近いものですが、激しい不連続性は回避できます。

構文形式は次の通りです。

```
value = SWAGE(expression, left_value, right_value, width )
```

これは次の表現

```
value = IF expression < 0 THEN left_value ELSE right_value
```

と等価ですが、唯一、遷移が width の幅をかけてスムーズに行われる点が異なります。

線形な形での遷移については RAMP 関数を参照ください。

Examples:

Samples | Misc | Swagetest.pde 参照（RAMP の用例含む）

3.6.11 VAL, EVAL 関数

選択された座標系において数式を評価するには 2 つの方法があります。

```
value = VAL(expression, x, y )
value = VAL(expression, x, y, z )
```

指定された座標点における数式 expression の値が計算されます。ただし、**座標変数の値は定数でなくてはなりません**。

この形式により節点の座標値と関数値との暗黙の対応付けが可能になります。値は求解プロセスの各フェーズで計算され記憶されるので、計算効率の向上につながります。

```
value = EVAL(expression, x, y )
value = EVAL(expression, x, y, z )
```

指定された座標点における数式 expression の値が計算されます。この場合、**座標変数の値は動的に変化するものでも構いません**。
この形式の場合には節点の座標値と関数値との暗黙の対応付けが行えません。従って参照の都度計算が行われるため、実行時間の増大につながります。

3.7 演算子

3.7.1 算術演算子

数式の記述の中で次の記号を使用することができます。

演算子	動作
-	単項演算子。符号を反転させる。
+	二項演算子。二項の和を求める。
-	二項演算子。二項の差を求める。
*	二項演算子。二項の積を求める。
/	二項演算子。二項の商を求める。
^	二項演算子。べき乗。
**	二項演算子。べき乗。^と等価。

3.7.2 関係演算子

条件文の中で次の演算子を使用することができます。

関係演算子

演算子	定義
=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

論理演算子

演算子	定義
AND	双方とも真
OR	少なくとも一方が真
NOT	否定（単項演算子）

アサインメント演算子

“=”は等号記号としての用途の他に、名称に対し値や数式をアサインする場合にも使用されます。

3.7.3 ストリング演算子

ストリング定数を構成するために次の演算子を使用することができます。

演算子	動作
+	2 つの文字列を結合する。

3.7.4 ベクトル演算子

次の演算子はベクトル量に対して種々の変換を行います。

3 次元の問題において、ベクトルはそれぞれの座標軸の方向ごとに一つずつ成分を持つものと仮定されます。

2 次元の問題において、ベクトルは 2 つの成分を持ち、その双方が problem の平面上に位置するものと仮定されます。外積や回転の場合のように、第 3 の成分がスカラーとして推論されることもあります。

`CROSS ( vector1, vector2 )`
: 2 つのベクトルから外積（cross product）を構成します。2 次元の場合には、problem の平面に垂直な方向への成分に等しいスカラー値が応答として返されます。

`DOT ( vector1, vector2 )`
: 2 つのベクトルの内積（dot product）を応答します。

`MAGNITUDE ( vector )`
: ベクトルの絶対値を応答します。

MAGNITUDE (argx, argy [, argz])
成分が argx, argy, argz（3 次元の場合）で与えられたときのベクトルの絶対値を応答します。

NORMAL (vector)
ベクトルの境界上での法線方向成分（スカラー）を応答します。*

NORMAL (argx, argy [, argz])
成分が argx, argy, argz（3 次元の場合）で与えられたときのベクトルの境界上での法線方向成分（スカラー）を応答します。*

TANGENTIAL (vector)
ベクトルの境界上での接線方向成分（スカラー）を応答します。*

TANGENTIAL (argx, argy [, argz])
成分が argx, argy, argz（3 次元の場合）で与えられたときのベクトルの境界上での接線方向成分（スカラー）を応答します。*

VECTOR (argx, argy [, argz])
与えられた引数を成分とするベクトルを構成します。

XCOMP (vector)
ベクトルの第 1 成分を応答として返します（座標軸の名前は x であるとは限りません）。

YCOMP (vector)
ベクトルの第 2 成分を応答として返します（座標軸の名前は y であるとは限りません）。

ZCOMP (vector)
ベクトルに第 3 成分がある場合に、それを応答として返します（座標軸の名前は z であるとは限りません）。

* 演算子 NORMAL と TANGENTIAL は境界条件定義の中で、あるいは境界プロットや境界積分の中でのみ使用できる点に注意。

3.7.5 微分演算子

微分演算子の名称には座標変数の名前が用いられます。それはデフォルトの名称である場合もあればユーザ定義による名称の場合もあります。

1 階微分用の演算子は“D<name>”という形を取ります。ここに <name> は座標変数の名前を意味します。

2 階微分用の演算子は“D<name1><name2>”という形を取ります。

2 次元直交座標系（Cartesian coordinates）の場合、デフォルトは“DX”、“DY”、“DXX”、“DXY”、“DYY”となります。

すべての微分演算子は problem で使用されている座標系に適合する形に内部で展開されます。

`D<n> ( arg )`
: arg の座標変数 <n> に関する 1 階の偏微分を表します。例: DX(arg)

`D<n><m> ( arg )`
: arg の座標変数 <n>, <m> に関する 2 階の偏微分を表します。例: DXY(arg)

`DIV ( arg )`
: ベクトル arg の発散（divergence）を計算します。

`DIV ( argx, argy {, argz } )`
: 成分が argx, argy, argz（3 次元の場合）で与えられたときのベクトルの発散（divergence）を計算します。

`GRAD ( arg )`
: スカラー arg の勾配（gradient）を計算します。

`CURL ( arg )`
: ベクトル arg の回転（curl）を計算します。3 次元の場合には回転ベクトルが応答となります。2 次元の場合には回転ベクトル（計算面に対して垂直）の絶対値に等しいスカラー値が応答となります。

`CURL ( scalar_arg )`
: スカラー scalar_arg の回転を計算します（2 次元の場合に限る）。scalar_arg は計算平面に垂直なベクトルの絶対値を表すものと仮定した上で、計算平面上のベクトルを応答します。

`CURL ( argx, argy {, argz } )`
: 成分が argx, argy, argz（3 次元の場合）で与えられたときのベクトルの回転（curl）を計算します。

`DEL2 ( scalar_arg )`
: scalar_arg のラプラシアン（Laplacian）を計算します。DIV(GRAD(arg)) と等価です。

3.7.6 積分演算子

立体、曲面、曲線上での積分を行うことができます。積分演算子に対する解釈は problem で使用されている座標系に依存します。積分演算子は引数としてスカラー関数のみをサポートしています。ベクトル場の積分は行えません。

Examples:

Samples | Steady_State | Heatflow | HeatBdry.pde
Samples | Misc | 3D_Integrals.pde
Samples | Misc | Constraints | Bdry_Constraint.pde
Samples | Misc | Constraints | 3D_Constraint.pde
Samples | Misc | Constraints | 3D_Surf_Constraint.pde
Samples | Misc | Tintegral.pde

(1) 時間積分

演算子 TINTEGRAL と TIME_INTEGRAL は同義であり、problem の開始時間から現在の時間に至るまで任意のスカラー値に対する時間積分を実行します。

```
TINTEGRAL ( integrand )
TIME_INTEGRAL ( integrand )
```

Note: この演算子は変数間に暗黙のリンクを形成する目的では使用できません。GLOBAL VARIABLE を使用してください。

(2) 線積分

演算子 BINTEGRAL と LINE_INTEGRAL は同義であり、線積分を実行します。積分は常に直線、または曲線上の距離に基づき行われます。

現状では線積分は 2 次元の problem でのみ利用できます。一般的な 3 次元での線積分については実装されていません。

2 次元直交座標においては LINE_INTEGRAL と SURF_INTEGRAL とは等価になります。2 次元円柱座標系の場合、SURF_INTEGRAL には重み $2\pi r$ がかかりますが、LINE_INTEGRAL にはかかりません。

```
BINTEGRAL ( integrand, named_boundary )
LINE_INTEGRAL ( integrand, named_boundary )
```

境界の指定は省略できます。その場合には外側の境界全体が仮定されます。

積分の対象となるリージョンを特定することもできます。

```
LINE_INTEGRAL ( integrand, named_boundary, named_region )
```

この場合、named_region は指定された境界によって仕切られるものでなくてはなりません。

(3) 2 次元体積積分

2 次元の場合の体積積分の指定様式は次の通りで、どちらを使用しても構いません。

```
INTEGRAL ( integrand, region )
VOL_INTEGRAL ( integrand, region )
```

ここで region の指定には番号、名称いずれも使用できます。また region の指定を省略することもできますが、その場合にはドメイン全体が仮定されます。

2 次元直交座標系の場合、体積要素は 2 次元のセルを 1 単位だけ z 軸方向に伸張する形で構成されるので、体積積分の値は座標平面上での面積分の結果と等しくなります。

2 次元円柱座標系の場合、体積要素は $2\pi rdrdz$ となるため、体積積分の結果と面積分の結果は一致しません。2 次元円柱座標系において

```
AREA_INTEGRAL ( integrand, region )
```

と指定した場合には、$2\pi r$ の重みがかからない形で面積分が実行されます（region の指定を省いた場合にはドメイン全体が対象となります)。

(4) 3 次元体積積分

3 次元の場合の体積積分の指定様式は次の通りで、どちらを使用しても構いません。

```
INTEGRAL ( integrand, region, layer )
VOL_INTEGRAL ( integrand, region, layer )
```

ここで layer の指定には番号、名称いずれも使用できます。また layer の指定を省略することもできますが、その場合にはレイヤスタック全体が仮定されます。
同様に region の指定には番号、名称いずれも使用できます。また region の指定を省略することもできますが、その場合には射出された平面（projection plane）全体が仮定されます。

region の指定が省略された場合には layer の指定は名称で行う必要があります。region, layer の双方が省略された場合にはドメイン全体が仮定されます。

例えば、

- INTEGRAL(integrand, region, layer) と指定した場合には、選択されたリージョンとレイヤに含まれる部分領域を対象に積分が行われます。
- INTEGRAL(integrand, named_layer) と指定した場合には、選択されたレイヤ上のすべてのリージョンを対象に積分が行われます。
- INTEGRAL(integrand, region) と指定した場合には、選択されたリージョンに属するすべてのレイヤを対象に積分が行われます。
- INTEGRAL(integrand) と指定した場合には、ドメイン全体を対象に積分が行われます。

(5) 2 次元面積分

2 次元の場合の面積分の指定様式は次の通りで、どちらを使用しても構いません。

```
SINTEGRAL ( integrand, named_boundary )
SURF_INTEGRAL ( integrand, named_boundary )
```

ここで named_boundary の指定には名称を用いますが、省略しても構いません。省略した場合にはドメインの外側の境界全体が仮定されます。

2 次元直交座標系の場合、面要素は 2 次元の線要素を 1 単位だけ z 軸方向に伸張する形で構成されるので、面要素は $1 \cdot dl$ となります。この場合、面積分の値は線積分の結果と等しくなります。
2 次元円柱座標系の場合、面要素は $2\pi r dl$ となるため、面積分の結果と線積分の結果は一致しません。

リージョンを特定する場合には第 3 の引数で次のように指定します。

```
SURF_INTEGRAL ( integrand, named_boundary, named_region )
```

この場合、named_region は指定された境界によって仕切られるものでなくてはなりません。

(6) 3 次元面積分

3 次元の問題の場合、面積分にはいくつかの形態があります。

1. Extrusion 面上での積分の場合には、曲面を名称または番号で選択し、それにリージョン（名称または番号）を修飾子の形で指定します。

   ```
   SINTEGRAL ( integrand, surface, region )
   SURF_INTEGRAL ( integrand, surface, region )
   ```

   region の指定が省略された場合には、指定された面上のすべてのリージョンを対象に積分が実行されます。
   surface と region の双方の指定が省略された場合には、ドメインの外側の面全体を対象に積分が実行されます。

   このタイプの積分に対してはさらにレイヤを特定することができます。

   ```
   SURF_INTEGRAL ( integrand, surface, region, layer )
   ```

   この場合、layer は指定された面によって仕切られるものでなくてはなりません。

2. “側壁”（sidewall）上の積分は境界名を選択し、それにレイヤ名を修飾子の形で指定します。

   ```
   SINTEGRAL ( integrand, named_boundary, named_layer )
   SURF_INTEGRAL ( integrand, named_boundary, named_layer )
   ```

   layer の指定が省略された場合には、指定された面上のすべてのレイヤを対象に積分が実行されます。

   このタイプの積分に対してはさらにリージョンを特定することができます。

   ```
   SURF_INTEGRAL( integrand, named_boundary, named_layer, named_region )
   ```

   この場合、named_region は指定された面によって仕切られるものでなくてはなりません。

3. 部分領域の境界面全体にわたる積分はリージョン名とレイヤ名を特定する形で指定します。

   ```
   SINTEGRAL ( integrand, named_region, named_layer )
   SURF_INTEGRAL ( integrand, named_region, named_layer )
   ```

   named_layer の指定が省略された場合には、指定された面上のすべてのレイヤを対象に積分が実行されます。

3.8 事前定義された要素

Problem descriptor の記述言語では次の要素をあらかじめ定義してあります。

```
PI          3.14159265358979
```

“R”を座標変数名とか他の定義名として設定していない直交座標系においては、problem descriptor 言語であらかじめ規定されている次の定義が有効になります。

```
R           R=SQRT(x^2 + y^2)          !  radius vector length in 2D
            R=SQRT(x^2 + y^2 + z^2)    !  radius vector length in 3D
THETA       THETA = ARCTAN(y/x)        !  azimuthal angle in 2D or 3D
```

Note: 数式中で使用される前に定義文の左辺で“R”や“Theta”が指定されると、それがこれらの名称に対する新たな意味となり、事前定義された内容は隠された形となります。

SELECT セクション中で“stages = integer”という指定のあるステージング機能を使用した problem に対しては、次の設定が有効になります。

```
STAGE       1 から integer までの整数値を取るインデックス
```

SELECT セクション中で“modes = integer”という指定のあるモード解析型の（固有値、固有関数）problem に対しては、次の設定が有効になります。

```
LAMBDA      固有値を表す名称
```

3.9 数式

一般の数式

Problem descriptor 上では一つ、または複数の演算子、変数、定義値、ペア括弧からなる数式を使用できます。ただしそれらは数値定数として評価されるものでなくてはなりません。数式の値を評価する場合、FlexPDE は代数のルールに則り、まず単項演算子を最初に評価し、次に 2 項演算子を次の順で評価します。

べき乗
乗除算
加減算
関係演算子（<, <=, =, <>, >=, >）
論理演算子（AND, OR）

ペア括弧で囲まれた部分数式（subexpressions）が数式中に含まれていた場合、それらは先行する、あるいは後続の演算子の優先順位に関係なく、最初に評価されます。括弧は何段にもネストされていて構いません。その場合、内側にある部分数式から順に評価が進行します。括弧はペアで使用されなくてはなりません。

条件付き数式

Problem descriptor の中では次の形の条件付き数式を使用できます。

```
IF condition THEN subexpression ELSE subexpression
```

この形式の場合、数式の値として 2 つのうちのいずれかが選択されます。一つの用例を示すと“y = IF a THEN b ELSE c”のようになります。

これはプロシジャ型の言語に良く見られる“IF a THEN y=b ELSE y=c”とは形が異なりますので注意してください。

THEN, あるいは ELSE に続く部分数式中には IF...THEN...ELSE がネストされて含まれていても構いません。それぞれの ELSE は直前に位置する IF と組み合わされます。

3.10 反復計算

REPEAT..ENDREPEAT construct を用いると反復計算が行えます。構文上はプロシジャ型言語における FOR ループに似ていますが、FlexPDE の場合はプロシジャの反復ではなくテキストの反復となる点に注意してください。

REPEAT クローズの形式は次の通りです。

```
REPEAT name = initial TO final
REPEAT name = initial BY delta TO final
```

これらのステートメントは後続のテキスト行を繰返し実行することを指示しています。指定された name は DEFINITIONS セクションで定義されたものと同様に扱われ、initial で指定された値がセットされます。

反復対象のテキスト行の末尾には次のステートメントを置きます。

```
ENDREPEAT
```

この時点で name には delta の値が加算されます（delta が指定されなかった場合には 1 が加算されます)。新たな値が final より大きくない場合には反復対象のテキスト行が再度スキャンされます。delta の値が負の場合には name に対して減算が行われ、また終了判定のロジックも変化します。

REPEAT ステートメントは次の場所で使用できます。

- BATCH ファイルリスト中
- VARIABLE リスト中
- EXTRUSION リスト中
- REGION, START, LINE キーワードが使えるところ
- プロットコマンド周辺
- DEFINITION 周辺
- REPORT コマンド周辺
- HISTORY リスト中の AT ポイント周辺

ARRAY と$integer ストリング機能を併用すると REPEAT ループの有効性がより高まります。

Examples:

```
REPEAT xc=1/4 by 1/4 to 7/4
  REPEAT yc=1/4 by 1/4 to 7/4
    START(xc+rad,yc) ARC(CENTER=xc,yc) ANGLE=360 CLOSE
  ENDREPEAT
ENDREPEAT
```

この 2 重のループにより 7×7 の円が形成されます。

用例については次を参照ください。

Samples|Misc|Repeat.pde
Samples|Misc|ArrayRepeat.pde

Note: REPEAT..ENDREPEAT は FlexPDE の初期バージョンで使用されていた FOR..ENDFOR を置き換えるものです。FOR..ENDFOR は互換性維持のためサポートはされていますが、新規に problem descriptor を作成する場合には使用しないでください。

第 4 章

セクション

Descriptor を構成する**セクション**の概要については第 2 章で説明しました。本章ではそれぞれのセクションにつき、その機能と内容を詳しく解説します。

4.1 Title

TITLE セクション（オプショナル）には一つの文字列を配置できます。

TITLE が指定された場合には、その文字列はすべての MONITORS、PLOTS 出力に対する見出しラベルとして使用されます。
TITLE が指定されなかった場合、プロット出力には見出しラベルは付加されません。

Example:

```
TITLE "this is my first model"
```

4.2 Select

SELECT セクション（オプショナル）は内部の**セレクタ**の値をデフォルトから変更する場合に使用します。これらのセレクタは処理の流れを制御するために用いられています。

SELECT セクションにおいては複数のセレクタとその値を設定できます。各種デフォルト値は FlexPDE ができるだけ広範な問題に対応できるようにという視点で設定されています。このデフォルトの振舞いが何らかの意味で不適切な場合にのみ設定を変更するようにしてください。

Program descriptor 中で使用される他の要素と異なり、セレクタとして用いられている名称は標準言語の一部ではありません。このため他のセクション中で使用しても意味をなしません。

FlexPDE 中で実装されているセレクタはバージョン固有であるため、FlexPDE の旧バージョンとか、FlexPDE の descriptor 言語を用いて作られている他のアプリケーションとは整合性が取れないことがあります。

4.2.1 メッシュ生成の制御

メッシュ生成のロジックを制御するために次のコントロールを SELECT セクション中で使用することができます。論理セレクタについては `selector = ON` と指定するか、単に `selector` を言及するだけでオンに設定できます。オフにするには `selector = OFF` と指定してください。

`ASPECT` default: `2.0`

2 次元及び 3 次元 surface mesh の生成においてセルのアスペクト比の最大値を規定します。セルは edge-size の比においてこの値まで引き伸ばせることになります。

`CURVEGRID` default: `On`

`ON` の場合、セルは曲線を描く境界に沿う形で生成されます。また 3D メッシュは曲面の曲率を表現できるように細分化されます。
`OFF` の場合、これらの修正は共に行われず、計算は辺が直線の三角形のまま、あるいは面が平面の四角錐のまま、進行することになります。（面が `TABLE` によって定義されている場合には、テーブル端で曲率が無限大となるため、このオプションを `OFF` にする必要が出てくることがあります。）

`GRIDARC` default: `30 degrees`

円弧をグリッド化する場合、どのセルもこの角度を超えないように制御します。他の要因によりセルのサイズはさらに小さくなることがあります。

`GRIDLIMIT` default: `8`

警告メッセージを出力することなく実行する再グリッド化（regridding）の最大試行回数。バッチ実行の場合はこの回数に達した時点で処理が打ち切られます。

`INITGRIDLIMIT` default: `5`

初期値を設定するために行う初期調整中に実行する regridding パスの最大回数。`INITGRIDLIMIT=0` と指定した場合には初期調整が抑止されます。

`NGRID`

それぞれの次元におけるメッシュの行数。開放領域における最大セルサイズを制御する上で使用できます。全体的なメッシュ密度を制御する上で簡便な方法を提供するものと言えます。デフォルト値は次の通りです。

	1D	2D	3D
Professional	100	15	10
Student	50	10	5

`NODELIMIT`

最大節点数。メッシュの細分化の過程でこの上限を越える節点が生成されようとした場合には、指定されたサイズのメッシュ全体で誤差をバランスさせるよう、セルの併合に関する閾値を増加させます。`NGRID`, ユーザによる密度制御、ドメイン境界のフィーチャサイズによって規定され生成される初期メッシュのサイズの制御には使用できません。デフォルト値は次の通りです。

	1D	2D	3D
Professional	2000000	2000000	2000000
Student	100	800	1600

`REGRID` default: `On`

デフォルトでは FlexPDE は adaptive な mesh refinement（動的メッシュ細分化）を実施します。このセレクタをオフにした場合にはその機能は抑止され、固定メッシュで計算が進行します。

`SMOOTHINIT` default: `On`

時間依存型の問題において、不連続な初期条件が与えられたときにその影響を軽減すべく、軽度な初期値の平滑化を実施します。

`STAGEGRID` default: `Off`

ステージングを使用した problem において各ステージごとにメッシュの再生成を強要します。FlexPDE はステージ間の依存関係を検出し、それに応じたメッシュ生成を行おうとするわけですが、その自動検出の機能を抑止する場合にこのセレクタは使用されます。

Note: メッシュ制御に関するより詳しい記述についてはサブセクション 4.6.7、及びユーザガイドの第 9 章“メッシュ密度の制御”を参照ください。

4.2.2 求解の制御

FlexPDE の求解プロセスを制御するものとして以下のコントロールを SELECT セクション中で使用することができます。論理セレクタについては selector = ON と指定するか、単に selector を言及するだけでオンに設定できます。オフにするには selector = OFF と指定してください。

AUTOSTAGE　　default: On

ステージング機能を使用する problem において、このセレクタがオンの場合にはすべてのステージが中断なく連続的に実行されます。このセレクタを OFF にすると FlexPDE の実行は各ステージが終わるごとに停止状態に置かれるので、結果を確認しながら先に進めることができます。

CHANGELIM　　default: 0.5（定常状態型）、2.0（時間依存型）

ニュートン法の反復ステップにおいて許されるノード変数値の変化（変数のノルムとの相対的尺度で計測）の最大値を規定します。非線形性の強い問題においては、非線形関数の病的な振舞いを回避するため、解に向かってゆっくり進むよう仕向けるといった措置が必要になることがあります。

CUBIC　　default: Off

有限要素法の基底関数（basis）として 3 次式を使用する（ORDER=3 と等価）。デフォルトは 2 次式（quadratic(ORDER=2)）です。3 次式の基底の場合、より多くのノードが生成されるため、ときにはシステムをよりたちの悪いものにしてしまう場合があります。

ERRLIM　　default: 0.002

演算精度にかかわるコントロールとして最も基本となるものです。空間的な誤差コントロール XERRLIM、及び時間的な誤差コントロール TERRLIM に対し、別個に値が明示されなければ、これらには ERRLIM で指定された値がセットされます。

[Note: ERRLIM は従属変数の相対誤差に対する**推定値**に過ぎません。得られた解がこの許容誤差範囲内にあることを保証するものではありません。望ましい解の精度に到達するためには ERRLIM の値を調整したり、メッシュ密度を高くするといった操作が必要になる場合があります。]

FIRSTPARTS　　default: Off

デフォルトでは FlexPDE はすべての 2 階微分項を部分積分し、自然境界条件によって表現される surface terms を生成します。このセレクタをオンにした場合には 1 階の微分項も同様に部分積分されることになります。なお、このオプションを使用した場合には、自然境界条件の記述に項の追加が必要になることがあります。

FIXDT default: Off

自動的なタイムステップコントロール機能を無効にします。この場合、タイムステップは TIME セクションで指定された値に固定されます。

HYSTERESIS default: 0.5

時間依存型の問題において空間誤差の推定が衰弱してしまうという問題に対しヒステリシスを導入します。この場合、一つ前の誤差推定値に指定された値をかけたものが現時点での推定値に足される形で実効的な誤差推定値が算定されます。多くの場合、regridding を安定化させる効果があります。

ICCG default: On

対称型の問題において不完全コレスキー共役勾配法（Incomplete Choleski Conjugate-Gradient）を使用します。この手法は通常、より迅速に収束します。ICCG=OFF の場合、または因子分解がうまく行かなかった場合には Orthomin 法が使用されます。

ITERATE default: 1000（定常状態型）、500（時間依存型）

共役勾配法における反復回数の第 1 次上限値。この値を越えると収束を強要するテクニックが適用されるようになります。反復回数のハードな上限値は 4*ITERATE となります。

LINUPDATE default: 5

線形の定常状態型問題においては、FlexPDE は残差が許容値より小さくなるまで線形系の求解を繰り返しますが、その反復回数の上限が LINUPDATE です。

MODES default: 0

固有値問題に対するソルバの選択を指示すると同時に、求めたいモードの数を規定します。デフォルトでは固有値問題用のソルバは起動されません。

NEWTON default: (5/changelim)+40

ニュートン法の反復回数の上限値を規定します。

NONLINEAR default: Automatic

ソルバ選択に関する FlexPDE の判断とは無関係に、非線形問題用のソルバ（Newton-Raphson）を選択します。

NONSYMMETRIC default: Automatic

ソルバ選択に関する FlexPDE の判断とは無関係に、非対称 Lanczos 共役勾配ソルバ（non-symmetric Lanczos conjugate gradient solver）を選択します。

NOTIFY_DONE　　default: Off

FlexPDE に対し、計算終了時にビープ音と“DONE”メッセージを発するよう指示します。

NRMATRIX　　default: 5

定常状態型の解法において、結合行列（coupling matrix）の再計算を行う前に Newton-Raphson の反復を最大何回行うかを規定します。この行列は解が感知できる程度に変化したり残差が大きくなった場合に再計算されます。

NRMINSTEP　　default: 0.009

Newton-Raphson backtracking 時に適用されるステップサイズに対する最小比率を規定します。この数値はむずかしい非線形系に対してのみ効果を発揮します。通常、計算されたステップサイズは変更されません。

NRSLOPE　　default: 0.1

定常状態型問題に対する Newton-Raphson backtracking において、残差の改善に関する最小許容値を規定します。

NRUPDATE　　default: 3

非線形の時間依存型問題において、それぞれのタイムステップ内で実行する Newton-Raphson の最大ステップ数を規定します。デフォルト値の 3 がコストと安定性の面でバランスの取れた値のように思われます。たちの良い非線形問題であれば、この値を 1 にすることによって処理時間の短縮が行えることもあります。このセレクタは PREFER_SPEED と PREFER_STABILITY によって自動的にセットされます。

NRUPFIT　　default: Off

非線形の時間依存型問題においてニュートン法の反復計算の都度、FIT と SAVE の値を再計算することを指示します。バージョン 2.20e 以前においてはこれらの値の計算は各タイムステップで 1 度だけ行われていました。デフォルトではタイムステップごとに 1 度のニュートン法のステップしか使用されないため、このセレクタの指定は NRUPDATE と組み合わせた場合にのみ意味を持つことになります。

ORDER　　default: 2

有限要素補間の次数を規定します（2 または 3）。セレクタ QUADRATIC と CUBIC はそれぞれ ORDER=2、ORDER=3 と等価です。

OVERSHOOT　　default: 0.001

収束制御用のパラメータです。共役勾配の解法は許容誤差が OVERSHOOT*ERRLIM となるまで反復を継続します。（アルゴリズムによってはさらなるパラメータを付加することがあります。）

PRECONDITION default: On

共役勾配の解法において行列の preconditioning を用いることを指示します。デフォルトの preconditioner は対角ブロックの逆行列です。

PREFER_SPEED default: Off

時間依存型の問題においてより高速の計算に重きを置いたパラメータ設定を行います。難しい非線形の問題に対しては PREFER_STABILITY の方を使うようにしてください。PREFER_SPEED は NRUPDATE=1, TNORM=2 と指定したのと等価です。

PREFER_STABILITY default: On

時間依存型の問題において計算時間はかかるがより安定性に重きを置いたパラメータ設定を行います。PREFER_STABILITY は NRUPDATE=3, TNORM=4 と指定したのと等価です。非線形問題でもたちの良いものであれば PREFER_SPEED を選択することにより計算時間を短縮できることがあります。

QUADRATIC default: On

有限要素法の基底関数（basis）として 2 次式を使用することを指示します。ORDER=2 と等価です。

REINITIALIZE default: Off

ステージングを用いた problem の各ステージにおいて、直前のステージの結果を保持するのではなく、INITIAL VALUES の指定に基づき毎回再初期化を行うよう指示します。

STAGES default: 1

ステージの数を選択することにより、パラメータ値を変化させたときの実行を自動化させることができます。各ステージでドメインの幾何形状パラメータに変化がない場合には、一つのステージのメッシュと解は次のステージにおける初期状態として使用されます。

SUBSPACE default: MIN(2*modes,modes+8)

固有値問題を選択すべく MODES が指定された場合、このセレクタは固有値計算で使用される部分空間（subspace）の次元を規定します。

TERRLIM default: 0.002

時間軸方向の計算精度にかかわるキーとなるコントロールです。時間依存型の問題の場合、時間積分における相対誤差推定値がこの値を超えたときにはタイムステップは半分にカットされます。逆に時間的誤差の推定値がこの値よりも小さくなった場合にはタイムステップの値は増やされます。TERRLIM の値は ERRLIM によって自動的にセットされます。

[Note: TERRLIM は従属変数の相対誤差に対する**推定値**に過ぎません。得られた解がこの許容誤差範囲内にあることを保証するものではありません。望ましい解の精度に到達するためには TERRLIM の値を調整するといった操作が必要になる場合があります。]

TNORM　　default: 4

時間依存型問題に対する誤差均等化プロセスに関連したパラメータです。タイムステップの制御はノード誤差の summed（2^TNORM）power に基づいています。許容される値は 1-4 の範囲です。大きなメッシュで局所的なアクティビティを伴う problem では TNORM として大きめの値を設定してください。

UPFACTOR　　default: 1

風上側拡散項に対する倍率です。この値を大きくするとわずかに双曲型の系を安定化できる場合があります。

UPWIND　　default: On

方程式の主変数中の風上側対流項（upwind convection terms）。対流項が存在した場合には流れの方向に拡散項が追加されたことになり、計算を安定化させる効果があります。

VANDENBERG　　default: Off

Vandenberg の共役勾配反復法（Vandenberg Conjugate-Gradient iteration）を使用します（双曲型の系が収束しなかった場合に有効です）。このアルゴリズムは $A\boldsymbol{x} = \boldsymbol{b}$ の代りに $({}^tAA)\boldsymbol{x} = {}^tA\boldsymbol{b}$ を解きます。これによって条件数（condition number）が 2 乗となるため収束は遅くなりますが、標準的な CG 法ではうまく行かないケースでも固有値をすべて正にするという効果を持ちます。

XERRLIM　　default: 0.002

空間的な計算精度にかかわるキーとなるコントロールです。従属変数中の空間的な相対誤差推定値がこの値を超えたときにはセルは分割されることになります（ただし NODELIMIT を越えない範囲で）。XERRLIM の値は ERRLIM によって自動的にセットされます。

[Note: XERRLIM は従属変数の相対誤差に対する**推定値**に過ぎません。得られた解がこの許容誤差範囲内にあることを保証するものではありません。望ましい解の精度に到達するためには XERRLIM の値を調整したり、メッシュ密度を高くするといった操作が必要になる場合があります。]

4.2.3 グローバルなグラフィックス制御

グラフィックスサブシステムの振舞いを変更するために以下のコントロールを SELECT セクション中で使用することができます。論理セレクタについては selector = ON と指定するか、単に selector を言及するだけでオンに設定できます。オフにするには selector = OFF と指定してください。

通常これらのセレクタは個々のプロットコマンド中での指定により上書きされます（サブセクション 4.15.2 を参照）。

ALIAS(coord) = "name"　default: Coordinate name
プロット軸ラベルに対する別名を規定します。

AUTOHIST　default: On
他に何のプロットも作成しない場合にもヒストリプロットの更新を行います。

BLACK　default: Off
すべてのグラフィックス出力をモノクロで行います。

CDFGRID　default: 51
CDF 出力グリッドのデフォルトサイズを規定します（51×51）。

CONTOURGRID　default: 51
等高線プロットの解像度を規定します。実際の計算セルサイズがこの解像度に対応したサイズを越えないのであれば、それらが用いられます。

CONTOURS　default: 15
等高線のレベル数の目標値。実際の作図に際しては適切な値が選択されるため、等高線の本数は必ずしも指定した通りになるとは限りません。

ELEVATIONGRID　default: 401
From..To 指定の elevation プロットにおいて使用されるグリッドサイズ。境界上の elevation の場合にはこの指定は無視され、実際のメッシュポイントが使用されます。

FEATUREPLOT　default: Off
このセレクタがオンの場合には FEATURE 境界をグレイでプロットします。これは 3.10b 以前のバージョンにおいてデフォルトの様式でした。

FINDERBINS　　default: 20

FlexPDE は計算メッシュ中におけるプロットポイントの検索、及び TRANSFER ファイル中における lookup ポイントの検索を高速に行うため、ドメインを含む長方形をいくつかの帯に分割します。局所的に高密度のメッシュを持った problem では、デフォルトの 20 バンドでは参照の効率化が十分に行えず、より多くのバンド数が必要になる場合があります。そのような場合にこのセレクタで値を設定します。[3.10a]

FONT　　default: 2

Font=1 の場合には sans-serif フォントが、Font=2 の場合には serif フォントが選択されます。

GRAY　　default: Off

カラーパレット（デフォルト）を用いるのではなく、すべてのプロットをグレイスケールで作成することを指示します。

HARDMONITOR　　default: Off

MONITORS の出力をハードコピー（.pg5）ファイルに書き出す場合に指定します。

LOGLIMIT　　default: 15

対数プロットの場合のデータの範囲は最大のデータ値から LOGLIMIT 桁下までに制限されます。これはグローバルな設定値であり、プロットコマンド上のローカルな LOG(number) 修飾子により上書きすることができます。

MERGE　　default: On

誤差の小さなメッシュセルの併合を許可します。それまでに分割されたセルのみが対象になります。

NOMINMAX　　default: Off

すべての等高線プロットより最大点、最小点を表す"x", "o"のマークを削除します。

NOTAGS　　default: Off

すべての等高線プロット、elevation プロットにおいてレベルを示すタグの出力を抑止します。

NOTIPS　　default: Off

ベクトルプロットにおいて矢印の頭の部分を書かないよう指示します。双方向のストレスをプロットする際に有用です。

PAINTED default: Off

カラーで塗りつぶした形の等高線プロットを作成します。プロットコマンド中で修飾子を用いることによりプロットごとに制御することもできます。

PAINTGRID default: On

カラーで塗りつぶしたグリッドプロットを作成します。カラーは材質ごとに別個の色が使用されます。

PAINTMATERIALS default: On

PAINTGRID と同義語です。プロットコマンドの修飾子との対称性を保つ意味で用意されています。

PAINTREGIONS default: Off

PAINTGRID をオンにしますが、カラースキームは異なります。カラーは材質パラメータごとにではなく、ロジカルリージョンごと（2D の場合)、あるいはロジカルコンパートメント（3D の場合）ごとに別個の色を使用します。

PLOTINTEGRATE default: On

すべての空間的プロットに対して積分値を計算します。デフォルトは体積積分と面積分であり、円柱座標系の場合には重み $2\pi r$ が用いられます。ヒストリプロットに対しては自動的には積分は計算されないので、必要な場合には積分式を明示する必要があります。

PRINTMERGE default: Off

それぞれの EXPORT ステートメントに含まれるすべてのステージとプロットタイムを一つのファイルに集約することを指示します。デフォルトの場合、EXPORT は時間ごと、あるいはステージごとに別個のファイルを生成します。プロットコマンド中での修飾子により EXPORT を個別に制御することもできます。

SURFACEGRID default: 51

Surface プロットに対する最小の解像度を規定します。

TEXTSIZE default: 35

プロット出力上の文字サイズを制御します。値は 1 ページ当りの行数を意味します。従って大きな値は小さな文字サイズを意味することになります。

THERMAL_COLORS default: On

プロットのラベリングにおいて使用されるカラーの順序を規定します。ON の場合には赤を一番上に配置します（高温に対応)。一方、OFF の場合には赤を一番下に配置します（スペクトル上の低周波に対応)。

VECTORGRID　　default: 41

ベクトルプロットの最小解像度を規定します。

VIEWPOINT(x,y,angle)　　default: negative X&Y, 30

SURFACE プロットに対する視点を規定します。角度の単位は度です。(3D の場合、これは切断面中の位置を規定することになります。)

4.3 Coordinates

COORDINATES セクション（オプショナル）によって problem に用いる座標系を規定することができます。基本的な構文は

```
COORDINATES geometry
```

ですが、geometry としては次のいずれかを指定します。

名称	意味
CARTESIAN1	1 次元直交座標系'X'
CYLINDER1	1 次元円柱座標系'R'
SPHERE1	1 次元球面座標'R'
CARTESIAN2	2 次元直交座標系'X', 'Y'
XCYLINDER	2 次元円柱座標系。この場合、円柱の軸方向を表す'Z' は横軸（X）方向に、動径方向を表す'R' は縦軸（Y）方向に配置される。
YCYLINDER	2 次元円柱座標系。この場合、動径方向を表す'R' は横軸（X）方向に、円柱の軸方向を表す'Z' は縦軸（Y）方向に配置される。
CARTESIAN3	3 次元直交座標系'X', 'Y', 'Z'

座標変数のリネーム

COORDINATES には座標変数を明示できる構文も用意されています。

```
COORDINATES geometry('Xname' ['Yname' ['Zname']])
```

この場合、引数'Xname' は横軸の座標変数名として、'Yname' は縦軸の座標変数名として使用されます。また'Zname' は extrude された座標軸に対する変数名として使用されます。これらの名称には引用符が付いていてもいなくても構いません。

座標変数の変更は微分演算子の再定義を伴います。例えば DX は D<Xname> のようになります。

DIV, GRAD, CURL 演算子は座標変数が明示された場合でも正しく展開されます。EQUATIONS セクション中でこれらの演算子を用いることによって問題の記述を大幅に簡略化することができます。

COORDINATES セクションが指定されなかった場合には CARTESIAN2 が仮定されます。

4.4 Variables

VARIABLES セクションでは problem descriptor 上で用いられるすべての従属変数を定義すると共に、それらに対し名称の設定を行います。VARIABLES セクションに現れるすべての名称に対し有限要素法による近似が行われます。それぞれの変数は problem のドメイン上で連続なスカラー場を規定するものと仮定されます。また各々の変数に対しては EQUATIONS セクション中で対応する偏微分方程式が規定されていることが仮定されます。

従属変数に名称を割り当てるに際し、次のようなルールが適用されます。

- 変数名はアルファベットで始まるものとします。数字や記号から始めることはできません。
- 変数名は 1 文字だけであっても構いませんが、時間変数として予約されている t は使用できません。
- 変数名の長さには制限はなく、任意の文字、数字、記号を組み合わせて使用することができます（予約語を除く）。
- 変数名中にセパレータが含まれていてはいけません。合成型の名称を設定する場合には'_'を使用してください（例えば temperature_celsius）。
- 変数名中で'-' は使用できません。これはマイナス記号を表す予約語です。

Example:

```
VARIABLES
  U,V
```

4.4.1 THRESHOLD クローズ

（従属）変数名に対しては THRESHOLD クローズ（オプショナル）を関連付けることができます。

THRESHOLD の値は FlexPDE が所定の演算精度（ERRLIM）を維持しなくてはならない変数の値に対する最小の範囲を規定します。言い換えるなら、THRESHOLD は解の詳細についてユーザが興味を失い始める変動幅を規定するものと言えます。

誤差の推定値は THRESHOLD の値と実際に観測された変数値の範囲との大きい方によってスケーリングされるので、problem ドメインにおける変数値の変動が THRESHOLD の値を越えた時点で THRESHOLD の値は意味を失います。かと言って THRESHOLD の値が大きすぎると解の精度が劣化し

ます。逆に小さすぎると必要もない計算精度を得るために多くの時間を浪費することになります。従って THRESHOLD を指定するのであれば、期待される変数値の範囲（最大値 − 最小値）に比べて適度な値を設定してください。

THRESHOLD クローズの形式には次の 2 種類があります。

```
variable_name ( THRESHOLD = number )
variable_name ( number )
```

Note: 多くの場合、THRESHOLD の使用は時間依存型問題、あるいは非線形の定常状態型問題において、初期値が一様な値を持つ場合、または解が一様な値に近づいて行く場合にのみ意味があります。

4.4.2 移動型メッシュ

FlexPDE v5 では計算メッシュを移動させるような設定を行うことができます。そのためには変更を加えたいそれぞれの座標変数に対し代替変数を割り当てます。指定方法は次のようになります。

```
variable_name = MOVE ( coordinate_name )
```

この宣言文によって座標変数 coordinate_name に対し代替変数 variable_name がアサインされることになります。その後、この代替変数に対し通常のように EQUATIONS や境界条件を設定することができます。これらの方程式や境界条件は計算ノードにおける座標変数の値に制約を課すことになります。

Example:

```
VARIABLES
  U,V
  Xm = MOVE(X)
```

4.4.3 SIMPLEX 修飾子

計算に使用される基底関数とは無関係に、ある変数を線形の基底関数でモデル化するよう仕向けることもできます。指定は次のように行います。

```
variable_name ( SIMPLEX )
```

特定の変数に対しより低次の基底関数を使用することによって計算の安定性を高められる場合があります。

4.5 Global Variables

GLOBAL VARIABLES セクションはフィールド変数と密接にリンクした補助的、あるいはサマリ的な値を定義する場合に使用します。

VARIABLE の場合にはノードごとに値を持った有限要素フィールドが表現されるのに対し、GLOBAL VARIABLE の方はドメイン全体で一つの値を持ちます。

GLOBAL VARIABLES は単なる DEFINITIONS とは異なります。DEFINITIONS の場合には参照に際して代数的な置換が行われるに過ぎません。これに対し GLOBAL VARIABLES にはマスター結合行列（master coupling matrix）の行と列がアサインされるため、有限要素方程式と同時に値が求められることになります。

GLOBAL VARIABLES セクションは VARIABLES セクションのすぐ後ろに続く形で記述します。GLOBAL VARIABLES の宣言に際して適用されるルールは VARIABLES に対するものと同一であり、また THRESHOLD の設定も可能です。

個々の GLOBAL VARIABLE は VARIABLES の場合と同一のルールによって EQUATIONS セクション中の一つのエントリと関連付けられます。

GLOBAL VARIABLES には境界条件は付帯しません。それらは定常状態型か時間依存型のいずれかであり、ドメイン上での積分によって定義されることもあれば、他の関数値によって定義されることもあります。

Examples:

Samples | Misc | Heaterssi.pde
Samples | Misc | Heaterti.pde 参照

Note: FlexPDE の旧バージョンではグローバル変数は SCALAR VARIABLES と呼ばれていました。この用法は互換性維持のため残されていますが、今後は新たな用語を使用するようにしてください。

4.6 Definitions

DEFINTIONS セクションは problem descriptor 上で用いる特定の定数や係数、関数を宣言し、それらに名称をアサインするために使用されます。

定義項目に対し名称をアサインする際のルールは次の通りです。

- 変数名はアルファベットで始まるものとします。数字や記号から始めることはできません。
- 変数名は 1 文字だけであっても構いませんが、時間変数として予約されている t は使用できません。
- 変数名の長さには制限はなく、任意の文字、数字、記号を組み合わせて使用することができます（予約語や座標変数名、変数名を除く）。
- 変数名中にセパレータが含まれていてはいけません。合成型の名称を設定する場合には'_'を使用してください（例えば temperature_celsius）。
- 変数名中で'-' は使用できません。これはマイナス記号を表す予約語です。

通常、定義項目を設定する場合、名称に続けてアサインメント演算子'=' を配置、さらに値、または数式を指定することによってその値が設定されます。定義項目はダイナミックな要素であり、値がアサインされたにしてもそれは初期値でしかありません。求解過程において必要に応じてその値を更新することができます。

Example:

```
Viscosity = 3.02e-4*exp(-5*Temp)
```

これらの定義項目は EQUATIONS セクションの偏微分方程式中ではインライン展開されます。それらは有限要素近似式上には表現されませんが、必要に応じて種々の時間と位置に対して計算されます。

リージョンパラメータの再定義

DEFINITIONS セクションで定義された名称に対しては、BOUNDARIES セクションの一部、あるいはすべての REGIONS においてその定義を上書きすることができます。この場合、その物理量はリージョンごとに異なる値を持つことになります。ある物理量の値が後続の REGIONS においてもれなく指定されるのであれば、DEFINITIONS セクション中での値の設定を省略することができます。

リージョン側でのパラメータ再定義の具体例についてはユーザガイドマニュアルを参照ください。

4.6.1 配列の定義

配列やデータリストに対し名称を定義することができます。ステートメント

```
name = ARRAY [ value_1 , value_2 ...  value_n ]
```

はそれぞれ値が value_1, ..., value_n である n 個の要素からなる配列として name を定義します。

後続のテキスト行中でこれらの値を参照するには次のように指定します。

```
name [ index ]
```

リスト中で指定された値はスカラー値として評価されるものでなくてはなりません。座標変数や従属変数への依存性が含まれていてはいけません。

Note: 旧バージョンにおいては配列要素はスクリプトを読み込んだ時点で計算可能な定数でなくてはなりませんでした。しかし v5 では使用される段階で計算可能であれば良い形に改められています。

Example:

```
definitions
xc=array(1/3, 2/3, 3/3, 4/3, 5/3)    { a list of X-coordinates }
yc=array(1/3, 2/3, 3/3, 4/3, 5/3)    { a list of Y-coordinates }
...
boundaries
  region 1
    repeat i=1 to 5             { an indexed loop on X-position }
      repeat j=1 to 5           { an indexed loop on Y-position }
        start(xc[i]+rad,yc[j])  { an array of circular dots at the }
        arc(center=xc[i],yc[j]) angle=360 { ...  tabulated coordinates }
        close
      endrepeat
    endrepeat
```

このスクリプトによってリージョン 1 内に 5 行 ×5 列の円が生成されます。他の用例については“Samples | Misc | arrayrepeat.pde”を参照ください。

4.6.2 パラメータ化された定義

定義文中にはプロシジャ型言語における定義文がそうであるように、最大 3 つまでの引数を設定することができます。パラメータ化された定義の構文は次の通りです。

```
name ( argname ) = expression
name ( argname1 , argname2 ) = expression
name ( argname1 , argname2 , argname3 ) = expression
```

この形は expression の中に引数（argname, argname1-3）に対する参照が含まれていて初めて意味を持ちます。このように定義された名称を参照する場合には引数に対する値を指定する必要があります。FlexPDE における他の定義と同様、これらのパラメータ値の指定には座標変数や従属変数に依存した数式を使用することができます。定義文中の引数（argname, argname1-3）はその中でのみ通用するローカルなものであり、該当する expression の外部においては未定義となります。

パラメータ化された定義文に対する引数として指定できる名称は座標変数や従属変数のような既知のものである必要はありません。グローバルに知られていない値であっても引き渡すことができます。

Note: この construct は関数参照の部分を定義文テキストで置き換える形で実装されています。プロシジャ型言語におけるようなランタイムコールの形式とは異なります。

Example:

```
DEFINITIONS
  uu(arg) = arg*arg
  ...
EQUATIONS
  div(a*grad(u)) + uu(u+1)*dx(u) +4 = 0;
```

この例の場合、方程式は次のように展開されることになります。

```
  div(a*grad(u)) + (u+1)*(u+1)*dx(u) + 4 = 0
```

“Samples|Misc|func.pde”についても参照ください。

4.6.3 ステージング

FlexPDE の**ステージング機能**を使用すると種々のパラメータ設定に伴う分析を自動化することができます。このモードの場合、FlexPDE は異なるパラメータ設定のもとで problem を何回も実行します。個々のステージの実行には一つ前のステージの解とメッシュが初期値として利用されます。

STAGES セレクタ

SELECT セクションにおいて

```
STAGES = number
```

のように指定すると problem は number 回だけ実行されるようになります。同時に STAGE という名前のパラメータが定義され、ステージ番号を保持するようになります。他の定義中でこのパラメータを使えば、ステージ番号とリンクしたパラメータ値を設定できます。次はその一例です。

```
Voltage = 100*stage
```

STAGED パラメータ

パラメータの定義を次のような形で行うこともできます。

```
param = STAGED ( value_1, value_2, ..., value_n )
```

この場合、パラメータ param はステージ 1 では値 value_1 を、ステージ 2 では値 value_2 を、といった具合にステージごとに異なる値を取るようになります。
STAGED パラメータが定義された場合には STAGES セレクタの指定はなくても構いません。STAGES セレクタの指定が省略された場合には STAGED リストの長さが実行すべきステージの数を規定することになります。STAGES セレクタの指定が行われた場合にはそちらの方が STAGED リストの長さに優先します。

用例については“Samples | Misc | Stages.pde”を参照ください。

ステージングと幾何形状

ドメインの幾何形状の定義の中でステージ化された物理量を参照している場合には解とメッシュは保持されず、ステージごとに新たな幾何形状に対応したメッシュが再生成されることになります。そのような場合でもヒストリプロットの表示は行えます。

用例については“Samples | Misc | Stage_Geom.pde”を参照ください。

FlexPDE はドメイン幾何形状のステージ依存性を検知しメッシュの自動再構築を試みます。しかし何らかの理由でこの依存性が検出されなかった場合には、グローバルセレクタ STAGEGRID を使ってグリッドのステージングを強制することができます。

4.6.4　座標点の定義

座標点に対し名称を付加することができます。

```
point_name = POINT(a,b)
```

この場合、a, b は定義の時点で定数値に評価されるものでなくてはなりません。従属変数や座標変数への依存性は許されませんが、ステージ番号への依存性は許されます。

座標点 (a,b) と書くべきところで代りにここで定義した名称を使うことができます。

4.6.5　データインポートの定義

(1) テーブル入力機能

FlexPDE にはテーブル形式のデータをインポートする機能が用意されています。

```
name = TABLE ( 'filename' )
```

このステートメントにより指定されたファイルからデータテーブルがインポートされ、それに対して定義された名称 name がアサインされます。

通常このステートメントはパラメータ定義（DEFINITIONS セクション中、あるいは REGION クローズ内におけるリージョン固有のパラメータ定義）の中に置かれ、テーブルデータにはその名称が対応付けられます。

TABLE は数式中でも使用できますが、その場合にはデータはその数式評価に固有のものとなります。

テーブルデータに対しては指定されたデータグリッド上で補間操作（linear, bilinear, trilinear）が行われます。

FlexPDE は外部のテーブルファイル（1/2/3 次元）を介して非解析的なデータをインポートしたりエクスポートしたりすることができます。この機能は実験的なデータが利用できる系をモデルする際に、あるいは他のプログラムとインタフェースを取る際に有用です。

インポート用のテーブルファイルは ASCII テキストファイルであり、どのテキストエディタでも作成できます。ユーザプログラムから作成することもできますが、FlexPDE 自身に出力させることも可能です（EXPORT 修飾子、または TABLE 出力文（MONITORS, PLOTS 参照））。

テーブル座標変数の変更

通常テーブルデータの座標変数の名称はテーブルファイル自身によって与えられます。しかし次の指定様式を使えばそれらをリネームできます。

```
name = TABLE ( 'filename', coord1 [,coord2...]  )
```

この場合、座標変数の名前は指定されたものによって置き換えられます。ただしこれらの名称は TABLE ステートメントで使用するより前に定義されていなくてはなりません。

パラメータ name が後続の計算中で使用された場合、その時点でのテーブル座標変数が補間計算に際して使用されます。例えばテーブル座標変数が空間座標 x, y であるとした場合、該当パラメータはテーブルデータが problem のドメイン上に展開されたとした上での値を取ることになります。

テーブル入力に関する別の指定様式については TABLEDEF ステートメントを参照ください。またテーブルデータの異なる補間方法については SPLINETABLE ステートメントを参照ください。

Examples:

```
Samples | Misc | Table.pde
```

(2) TABLEDEF 入力文

テーブル形式のデータをインポートする場合には TABLEDEF というステートメントを用いることもできます。

```
TABLEDEF('filename',name1 [,name2,..])
```

TABLE ステートメントの場合と異なり、TABLEDEF ステートメントは多値のテーブルファイルから直接、一つ、または複数のパラメータを規定できます。TABLE ステートメントの場合、追加の引数を用いて座標変数の割当て変更が行えたわけですが、TABLEDEF ステートメントの場合の引数は定義されるべき、そしてテーブルデータと対応付けられるべき名称を意味します。この点において TABLEDEF ステートメントは構文的に TRANSFER ステートメントと同一であると言えます。

テーブルファイルの形式については (4) を参照ください。

(3) SPLINETABLE 機能

SPLINETABLE は TABLE 入力機能の一変形であり、同じテーブル形式を使用します。

```
name = SPLINETABLE ( 'filename' )
```

SPLINETABLE の場合、補間計算には 3 次スプラインが用いられます。この場合、テーブルの格子点において、関数値のみならず、その 1 次導関数、2 次導関数もすべて連続になるよう補間関数が構成されます。なお、テーブルの周縁においては曲率 0 という条件が適用されます。

現時点でサポートされているのは 1 次元と 2 次元のテーブルのみです。

Examples:

```
Samples | Misc | Spline1.pde
Samples | Misc | Splinetable.pde
```

(4) TABLE ファイル形式

TABLE, TABLEDEF, SPLINETABLE 入力機能で使用するテーブルは次のような形式でなくてはなりません。

```
{ comments }
name_coord1 datacount1
  value1_coord1 value2_coord1 value3_coord1 ...
name_coord2 datacount2
  value1_coord2 value2_coord2 value3_coord1 ...
name_coord3 datacount3
  value1_coord3 value2_coord3 value3_coord3 ...
data { comments }
data111 data211 data311 ...
data121 data221 data321 ...
data131 data231 data331 ...
  ...     ...     ...
  ...     ...     ...
data112 data212 data312 ...
data122 data222 data322 ...
data132 data232 data332 ...
  ...     ...     ...
  ...     ...     ...
```

ここに

name_coordN	方向が N の座標変数名。典型的には name_coord1 が x、name_coord2 が y です。
datacountN	N 方向におけるデータ点の数。
DataJKL	座標点 (J,K,L) におけるデータ。

Example:

```
{ this is an example table }
x 6
  -0.01 2 4 6 8 10.01
y 6
  -0.01 2 4 6 8 10.01
data
  1000 1    1000 1    1000 1
  1    1000 1    1000 1    1000
  1000 1    1000 1    1000 1
  1    1000 1    1000 1    1000
  1000 1    1000 1    1000 1
  1    1000 1    1000 1    1000
```

(5) TRANSFER 入力文

TRANSFER ステートメントは FlexPDE 間でのデータ転送を可能にします。この場合、実行状態に関する完全な情報が維持されます。

```
TRANSFER ( 'filename', name1 [,name2, ...]  )
```

TRANSFER 入力機能で指定されるファイルは TRANSFER 出力機能によって書かれたものでなくてはなりません。入力機能で指定された名称は“name=”という定義文で宣言されたかのように扱われます。これらの名称はその順番に出力ファイル中のデータフィールドと対応付けられます。

Examples:

Samples | Misc | Import-Export | Transfer_Out.pde

Samples | Misc | Import-Export | Transfer_In.pde

(6) TRANSFERMESH 入力文

TRANSFERMESH 入力文の形式は TRANSFER ステートメントのそれと同一です。

```
TRANSFERMESH ( 'filename', name1 [,name2,..]  )
```

しかし TRANSFERMESH 入力ステートメントはディスク上のデータ定義情報をインポートするだけでなく、インポートされたファイルの有限要素メッシュ構造を現行の problem に対しても適用する点が異なります。このため通常のメッシュ生成のプロセスはバイパスされることになります。

このプロセスがうまく機能するためには、インポートする側のドメイン定義文とエクスポートする側のそれが完全に同一である必要があります。
TRANSFERMESH ファイルにはエクスポート側 problem の境界や境界条件に関する完全な記述が含まれているわけではなく、単にメッシュのレイアウト情報が含まれているに過ぎません。このためエクスポート側のドメイン定義をインポート側の descriptor にコピーしておく必要があります。境界条件は変更しても構いませんが、境界の位置と順番は変更してはなりません。

TRANSFERMESH は TRANSFER 出力文で生成されたファイルのみを読むことができます。

Examples:

```
Samples | Misc | Import-Export | Mesh_out.pde
Samples | Misc | Import-Export | Mesh_in.pde
```

(7) TRANSFER ファイル形式

TRANSFER ファイルの形式は TRANSFER 出力機能によって規定され、次のようなデータを含んでいます。

1) ヘッダ。ヘッダ中には FlexPDE のバージョン、出力側 problem の名称と実行時間を識別するためのセクション、及びプロットされた変数名、または関数式が含まれています。このヘッダは {...} というコメントの形式になっています。
2) ファイル識別子"FlexPDE transfer file"、及び problem のタイトル。
3) 幾何学的次元の数とその名称。

4) 有限要素基底関数の ID（4-10）。
   - 4 = 1 次の三角形（linear triangle, 3 点/セル）
   - 5 = 2 次の三角形（quadratic triangle, 6 点/セル）
   - 6 = 3 次の三角形（cubic triangle, 9 点/セル）
   - 7 = 3 次の三角形（cubic triangle, 10 点/セル）
   - 8 = 1 次の四面体（linear tetrahedron, 4 点/セル）
   - 9 = 2 次の四面体（quadratic tetrahedron, 10 点/セル）
   - 10 = 3 次の四面体（cubic tetrahedron, 20 点/セル）

5) 自由度の数（上記セル当りの点数）

6) 出力された変数の数とその名称

エクスポート側の problem 中に存在する異なった材質は TRANSFER ファイル上でそれぞれ別個のセクションによって表現されます。個々のセクションは次の情報から構成されます。

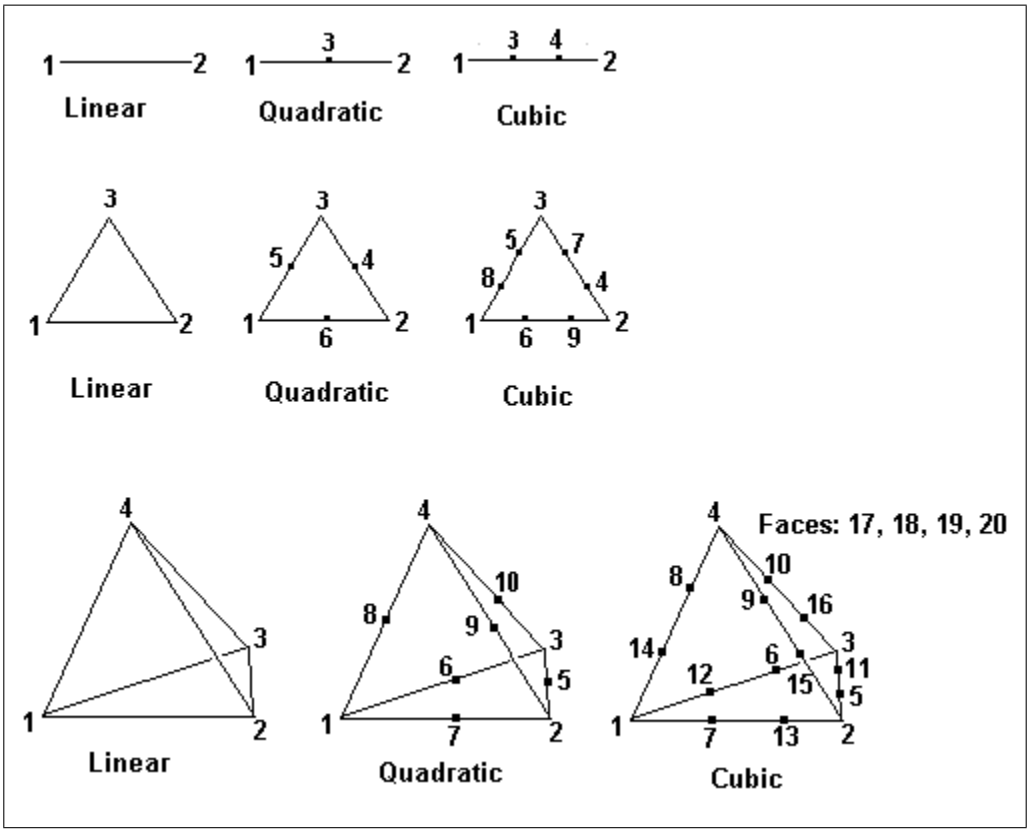
7) ノードの数。

8) ノードデータ。各ノードごとに次の情報を含む別個の行が生成されます。
   - 2 つ、または 3 つの座標値、及び (6) で規定される数だけのデータ値
   - コロン（:）
   - グローバルなノード番号（node index）
   - ノードタイプ（0=interior; 1=joint; 2=edge; 3=face; 4=exterior）
   - タイプ修飾子（region number, joint number, edge number, face number）

9) セルの個数

10) セルの結合性に関するデータ。各セルごとに次の情報を含む別個の行が生成されます。
   - 基底関数種別（(4) 参照）
   - セルを構成するノード番号（材質ブロック内ローカル）。これらノード番号の数は (5) で規定されます。
   - コロン（:）
   - グローバルなセル番号
   - ロジカルなリージョン番号
   - 材質番号

ノード番号を振る順番は次の図の通りです。



4.6.6　PASSIVE 修飾子

定義項目に対し PASSIVE という指定を行うことができます。この場合、それらの項目に対してはグローバルなヤコビ行列 (Jacobian matrix) を作成する際、システム変数による微分が抑止されます。非線形性の強い系の場合、この措置によって病的な振舞いが抑制されることがあります。ただし収束が遅くなるというマイナス面もあります。

Example:

```
Viscosity = Passive(3.02*exp(-5*Temp))
```

この場合、Viscosity を Temp で微分した結果は (-5)*3.02*exp(-5*Temp) とならずに 0 となります。

4.6.7 メッシュ制御パラメータ

MESH_SPACING と MESH_DENSITY という名称はメッシュの初期レイアウトを制御するための特別な意味を持っています。これらはパラメータの定義というコンテクストの他に、その再定義（DEFINITIONS セクション、または REGION 内）、あるいは境界条件のコンテクストにおいて使用できます。

MESH_SPACING はメッシュノード間の望ましい間隔を指定します。一方、MESH_DENSITY は MESH_SPACING の逆数であり、単位長あたりの望ましいノード数を指定します。

DEFINITIONS セクション内に現れた場合、これらのパラメータはドメインに対応した容量中におけるメッシュ密度関数のグローバルなデフォルト値を規定することになります。

REGION 中に置かれた場合には、これらのパラメータは該当リージョンに対応した容量中におけるメッシュ密度関数を規定します（3D の場合、それらには LAYER や SURFACE という修飾子が付くことがあります）。

境界条件というコンテクスト（すなわちパス内）で指定された場合、それらは該当する曲線や側壁面に沿ったメッシュ密度を規定することになります。3D の場合には密度関数の適用範囲を制限するため、LAYER や SURFACE という修飾子が付けられることがあります。

MESH_SPACING と MESH_DENSITY に対しては空間座標変数で表現された任意の関数を指定できます（従属変数を含めることはできません）。

メッシュ上の一部において複数の規定が有効だった場合には、最小のメッシュセルを生むコントロールが使用されます。

Example:

```
MESH_DENSITY = exp(-(x^2+y^2+z^2))
```

この指定の場合、$(0, 0, 0)$ の周囲の正規分布に従う密度分布が設定されます。ただしその間隔は最終的には NGRID で規定されるサイズの上限によって規制を受けることになります。

メッシュ密度の制御についてはユーザガイドの該当する章、及び次のサンプルコードを参照ください。

Examples:

Samples | Misc | Mesh_Control | Mesh_Density.pde
Samples | Misc | Mesh_Control | Mesh_Spacing.pde
Samples | Misc | Mesh_Control | Bdry_Density.pde
Samples | Misc | Mesh_Control | Bdry_Spacing.pde

4.7 Initial Values

INITIAL VALUES セクションは従属変数を初期化する場合に使用されます。特に初期化を行わなかった場合、従属変数は 0 に初期化されます。

定常状態型の問題の場合、INITIAL VALUES セクションはオプショナルです。

時間依存型の問題の場合には、それぞれの従属変数に対する値の代入文が INITIAL VALUES セクション内に用意されるべきです。初期値の設定に際しては従属変数名をアサインメント演算子“=”の左辺に置き、右辺には定数や関数、数式、あるいは定義済みの項目を配置します。

Example:

```
INITIAL VALUES
  U = 1.0-x
```

テーブルによる初期値設定

インポートされたテーブルから直接初期値を設定することもできます。

```
INITIAL VALUES
  U = TABLE("initial_U.tbl")
```

構文上の理由から TRANSFER (q.v.) から直接初期値を設定することはできません。TRANSFER コマンドで一旦中間的な名称を定義した上で、それを使って初期化を行ってください。

```
DEFINITIONS
  TRANSFER("initial_U.xfr",U0)
INITIAL VALUES
  U = U0
```

4.8 Equations

EQUATIONS セクションは problem の従属変数を定義する偏微分方程式を一覧として記述するために使用されます。VARIABLES セクション中にリストされた各々の従属変数に対し一つの方程式が与えられなくてはなりません。

方程式は紙の上に書くのと近い形で problem descriptor 上に記述できます。DIV(divergence), GRAD(gradient), CURL, DEL2(Laplacian) といった演算子を利用すると偏微分方程式が簡潔に表現できます。FlexPDE はこれらの演算子を COORDINATES セクション中で指定された座標変数による表現に正しく展開します。

偏微分項の入力が必要な場合には D<name> や D<name1><name2> といった形式の微分演算子を使用してください。ここで <name> は X, Y, Z といった（あるいは COORDINATES セクションでユーザによって明示された）座標変数の名称を意味します。

デフォルトの 2D 直交座標系においては演算子 DX, DY, DXX, DXY, DYX, DYY が定義されています。

同様にデフォルトの円柱座標系の場合 (XCYLINDER, YCYLINDER)、演算子 DR, DZ, DRR, DRZ, DZR, DZZ が定義されています。

3D 直交座標系の場合、演算子 DZ, DZZ, DXZ, DYZ が定義されています。

Example:

```
div(k*grad(u)) + u*dx(u) = 0
```

3 階以上の導関数

方程式の記述においては 1 階と 2 階の導関数のみが入力できます。より高階の導関数を必要とする重調和方程式 (biharmonic equation) のような場合には、中間変数を用いることにより、1 階と 2 階の導関数のみで記述される形に書き換える必要があります。

4.8.1 方程式、変数、境界条件の対応

従属変数が一つしかない問題の場合には境界条件と方程式の対応付けについてあいまいさはありません。

従属変数が複数ある問題の場合には、それぞれの方程式に変数名を付すことによってそれらの対応付けが行われます。このプロセスは結合行列中における方程式配列の最適化にも利用されます。

Example:

```
U: div(k*grad(u))+u*dx(u)= 0
```
（方程式を変数 U に対応付けます）

境界条件は BOUNDARIES セクション中で規定されますが、それらは変数名を介して方程式と対応付けられます。例えば VALUE(U)=0 と設定した場合、選択された境界上の節点における U に対する方程式は等式 u=0 によって置き換えられます。

自然境界条件は生成項が左辺に移項されたときのサインを用いて記述されなくてはなりません。自然境界条件の符号に伴う混乱を回避するためにも、2 階の微分項はすべて左辺に記述するようにしてください。

4.8.2 モード解析と方程式

モード解析を行う場合には SELECT セクション中で次のセレクタの宣言を行う必要があります。

```
MODES = integer
```

ここに integer は解析対象のモードの数を表します。

この場合、方程式は次のような形式で記述します。

```
F(V) +LAMBDA*G(V) = H(X,Y)
```

ここで F(V) と G(V) は従属変数を含む適切な項であり、H(X,Y) は driving source を表す項です。名称 LAMBDA は固有値を表すものとして自動的に宣言されるため、DEFINITIONS セクション中で宣言することはできません。

4.8.3 メッシュの移動

FlexPDE v5 ではメッシュ移動の機能が加わりました。この機能を使用するためには次のような設定を行う必要があります。

- 移動させたい座標変数に対する代替変数 (surrogate variables) のアサインメント
- それぞれの代替座標変数ごとに動きを規定する EQUATION の定義
- 代替座標系における境界条件

通常はメッシュ速度を規定する変数が設定されます。これは流れの速度と同じであっても良いですし（この場合、モデルは純粋にラグランジュ型になります)、その他のより良性の動きを表すものであっても構いません（この場合にはモデルはラグランジュ/オイラー混合型 (ALE) となります)。

FlexPDE v5 には歪んだメッシュを再結合する機能は含まれていません。純粋なラグランジュ型モデルの場合、重度のメッシュ破壊が起こる可能性があるため、なるべく使用しないようにしてください。

内部的なメッシュ再分散

便利なテクニックは座標変数ごとにメッシュ速度を表す変数を定義し、境界条件もそれに合わせて移動させる方法です。内部的には速度を

```
DIV(GRAD(x_velocity)) = 0
```

に従って拡散させることができます。なお、速度は次のようにして代替変数と対応付けます。

```
VELOCITY(x_surrogate) = x_velocity
```

メッシュ移動の影響

EQUATIONS はメッシュを移動させるか否かにかかわらず、常にオイラー（実験室）座標系に基づいて記述します。FlexPDE はメッシュ移動に伴う補正項を自動的に計算します。

4.9 Constraints

CONSTRAINTS セクション（オプショナル）は系に対し積分に関する条件を課する場合に使用します。これらの制約条件は機械的、化学的反応系のような定常状態系におけるあいまいさ、あるいは微分境界条件しか指定されなかった場合のあいまいさをなくす上で使用できます。

CONSTRAINTS セクションには通常次の形のステートメントを配置します（複数設定可）。

```
INTEGRAL ( argument ) = expression
```

CONSTRAINTS は境界条件によってあいまいさを持たない形で規定された定常状態系、あるいは時間依存系に対しては使用しないでください。

一つの CONSTRAINT は求解過程で最小化されることになる新たな付帯汎関数を生成します。CONSTRAINT による要件と PDE、あるいは境界条件が規定する要件との間にコンフリクトが生じた場合には、最終的な解はこれらの要件の**妥協**の産物となり、厳密にはそのいずれをも満足しないものとなる可能性があります。

CONSTRAINTS はどの INTEGRAL 演算子に対しても設定できます。

CONSTRAINTS は正値性等のローカルな要件をノード変数に強いる目的では使用できません。

Examples:

Samples | Misc | Constraints | Constraint.pde
Samples | Misc | Constraints | Bdry_Constraint.pde
Samples | Misc | Constraints | 3D_Constraint.pde
Samples | Misc | Constraints | 3D_Surf_Constraint.pde
Samples | Steady_State | Chemistry | Reaction.pde

4.10 Extrusion

3 次元の問題におけるレイヤ構造は EXTRUSION セクションにおいて下から順に指定して行きます。

```
EXTRUSION
  SURFACE "<Surface_name_1>" Z = expression_1
    LAYER "<Layer_name_1>"
  SURFACE "<Surface_name_2>" Z = expression_2
    LAYER "<Layer_name_2>"
  ...
  SURFACE "<Surface_name_n>" Z = expression_n
```

指定は SURFACE で始まり SURFACE で終わるものでなくてはなりません。
LAYER はこれらの面ではさまれた空間に対応します。これらのレイヤを参照するための名称が特に必要ないのであれば LAYER の指定は省略できます。

- 面は平面でなくても構いません。併合は許されますが交差は許されません。expression_1 は expression_2 以下にあるすべての領域とみなされ、他も同様です。交差の可能性がある場合には MIN, あるいは MAX 関数を使用するようにしてください。
- 面を規定する数式中ではリージョン別に定義されたパラメータを参照することができます。これによって面の定義をリージョンごとに別個のものとすることが可能です。しかしそれらばらばらの数式であってもリージョン境界においては連続につながっていなくてはなりません。具体例については“Samples | Misc | 3d_Domains | Regional_surfaces.pde”を参照ください。
- 面の数式中に条件付きの値が含まれている場合（IF...THEN や MIN, MAX, 等)、gridder によって識別可能であるためには、その区分 (breaks) に対応した FEATURE が基盤面のドメイン中に含まれていなくてはなりません。
- 面は材質境界も含め、あらゆる場所において連続でなくてはなりません。条件式やリージョン別の定義を行った場合には特に注意が必要です。

- 面を表す数式中でテーブル形式の入力データを参照することができます。具体例については“Samples | Misc | 3D_Domains | Tabular_surfaces.pde”を参照ください。

これらの指定においてレイヤや面に対する名称は省略できます。レイヤに対し名称を設定しない場合には LAYER ステートメント自体も省略できます。

短縮形

ラベルを付けない場合には EXTRUSION の指定を次のような短縮形で行うことができます。

```
EXTRUSION Z = expression_1, expression_2 [, ...]
```

この形式を使用した場合、面やレイヤの参照は番号で行われることになります（面に対しては 1 から n までの、レイヤに対しては 1 から $(n-1)$ までの番号が振られます)。SURFACE #1 は Z=expression_1 を意味し、LAYER #1 は SURFACE #1 と SURFACE #2 とではさまれたレイヤを意味することになります。

押出し (extrusion) に関するさらなる情報についてはユーザガイドを参照ください。

4.11 Boundaries

BOUNDARIES セクションは求解対象のドメインを規定すると共に、ドメインの外側の表面に沿った境界条件の設定に使用されます。

FlexPDE においては過去からの経緯もあって、境界に関する議論は 2 次元の指向性の強いものとなっています。3 次元の形状は 2 次元のドメインを第 3 の次元に押し出す (extrude) 形で構成されます。一方、1 次元のドメインは 2 次元のテクニックを特化する形で構成されます。

Problem descriptor 中には一つの BOUNDARIES セクションが存在していなくてはなりません。

Problem の境界定義は 2 次元直交座標系において、個々の材質で規定されるリージョンの外周をなぞることによって行われます。

このようにして物理的なドメインは REGION, FEATURE, EXCLUDE というサブセクションに分解されることになります。

Problem descriptor 中には少なくとも一つの REGION サブセクションが存在していなくてはなりません。FEATURE, EXCLUDE サブセクションの方はなくても構いません。

これらの構成要素に関する具体例については FlexPDE と共に配布されるサンプル problems を参照ください。

4.11.1 点

Problem のドメインを構成する際に使用される基本的な単位は幾何学的な点 POINT です。FlexPDE スクリプト上、点は座標値を括弧で囲む形で表記されます。2 次元の点であれば (2.4, 3.72) のような形になります。

2 次元、3 次元形状共に 2 次元のレイアウトからスタートしているため、3 次元の点の指定は一般的には ELEVATION PLOTS に限定されます。

1 次元の系の場合、座標値は一つのみとなるため、(2.4) のような表記となります。

4.11.2 境界パス

境界パスの一般的な指定形式は次のようになります。

```
START(a,b) segment TO (c,d) ...
```

ここで (a,b), (c,d) は segment の両端の座標を、segment は LINE, SPLINE, ARC のいずれかを意味します。

パスは segment をつなぎ合わせる形で構成されます。その場合、一つの segment の終端は次の segment の始点になります。

次に続くものが segment と解釈できなかった場合、あるいは START ポイントに戻り閉ループとなった場合に一つのパスは終結します。パスを閉じさせるには segment を START ポイントで終わらせるか、CLOSE を明示することによって行います。CLOSE を指定した場合には START ポイントまでパスが延長されます。

```
... segment TO CLOSE
```

または

```
... segment CLOSE
```

Note: 旧バージョンではパスを閉じさせるのに FINISH という単語を用いていました。FlexPDE v5 でもエラーにはなりませんが、誤解を招きやすい表現なので極力使わないようにしてください。

直線セグメント

直線のセグメントは次の形で指定します。

```
LINE TO (x,y)
```

続けて直線のセグメントを設定する場合には LINE という語を省略しても構いません。

```
LINE TO (x1,y1) TO (x2,y2) TO (x3,y3) TO ...
```

スプラインセグメント

スプラインセグメントの設定は構文的には直線セグメントの場合と同様に行えます。

```
SPLINE TO (x,y) TO (x2,y2) TO (x3,y3) TO ...
```

指定された点にフィットする形で 3 次スプライン曲線が生成されます。スプラインの始点は START ポイント、または直前のセグメントの終端に置かれます。スプラインの終点は TO(,) のチェインで指定した最後の点になります。
スプラインは両端の点においてその曲率が 0 となるよう設定されます。従って適切な向きでスプラインが終わるよう、始点と終点の周囲では密に点を設定する配慮が必要になります。

円弧セグメント

円弧セグメントにより円弧、または楕円弧を設定することができます。指定方法には次のように何種類かのパターンがあります。

```
ARC TO (x1,y1) to (x2,y2)
ARC ( RADIUS = R ) to (x,y)
ARC ( CENTER = x1,y1 ) to (x2,y2)
ARC ( CENTER = x1,y1 ) ANGLE=angle
```

ここに angle は度を単位とした角度を表します。標準的な慣例に従い、正の値は反時計回りの, 負の値は時計回りの角度を意味します。弧の終端の座標値は半径を指定された角度だけ回転させることによって決定されます。角度をラジアンで指定する場合にはラジアンの値の後に RADIANS という修飾子を付けてください。

ARC (CENTER=x1,y1) to (x2,y2) という形式が使用され、かつ中心 (x1,y1) が始点と終点から等距離にない場合、長軸、短軸が x 軸、y 軸に平行な楕円弧が生成されます。

Example:

```
START(0,0)
LINE TO (10,0) TO (10,10) TO (0,10) TO CLOSE
```

名前付きのパス

パスに名称を設定することができます。名前をパスにアサインする場合には、名称は引用符で囲まれた文字列で指定し、予約語 START の直後に配置します。

```
START "namedpath" ( <x> , <y> )
```

境界や直線に沿った積分を計算したい場合、あるいは ELEVATION プロット用のパスを設定する際に、名前付きのパスは有用です。

4.11.3　リージョン

REGION は単一の材質からなる領域を囲む境界パスによって仕切られた 2 次元 problem ドメイン（または 3 次元 problem ドメインの射影）の一部です（例外については後述する“1 次元のリージョン”の項を参照ください）。

REGION 内の材質の特性はどんなに複雑なものであれ、一つの定義によって規定されます。

REGION は複数の切り離された領域から構成される場合もあります。

Example:

```
REGION 1   { an outer box }
  START(0,0)
  LINE TO (10,0) TO (10,10) TO (0,10) TO CLOSE

REGION 2   { with two embedded boxes }
  START(1,1)
  LINE TO (2,1) TO (2,2) TO (1,2) TO CLOSE
  START(5,5)
  LINE TO (6,5) TO (6,6) TO (5,6) TO CLOSE
```

リージョンの重ね合わせ

後に定義されたリージョンはそれより前に定義されていたものを隠す形で、その上に重ねて配置されます。2 つのリージョンに共通の領域は後に定義されたリージョン内にのみ存在します。

上の例の場合には、2 つの小さな矩形領域が大きな矩形領域の上に重なって配置されます。大きな矩形領域に対しアサインされた材質パラメータは小さな矩形によって覆われていない領域でのみ意味を持ちます。

通常、最初のリージョンは problem のドメイン全体をカバーするように設定されます。次にこのデフォルトリージョンとパラメータの異なる部分領域に対しリージョンを重ねて設定します。ドメインのすべてをこのようなサブリージョンで覆ってしまった場合には、ドメイン全体をカバーする最初のリージョンは見えなくなります。このような設定は境界条件の指定をローカライズする上で有効なことがあります。とは言え、サブリージョンの一つは余計であると言えます。なぜならそれはデフォルトリージョンでも良いはずだからです。

(1) リージョナルパラメータの設定

DEFINITIONS セクション内で定義済みの名称に対し、次のようなアサイン文を予約語 REGION の直後に配置することにより、REGION 内でのみ有効な新たな値を設定することができます。

```
name = new_expression
```

このように定義項目に対し新たな値が再設定された場合、その値は該当リージョン内でのみ有効となります。

Example:

```
DEFINITIONS
  K = 1  { the default value }
REGION 1  { assumes default, since no override is given }
  START(0,0) LINE TO (10,0) TO (10,10) TO (0,10) TO CLOSE
REGION 2
  K = 2  { both sub-boxes are assigned K=2 }
  START(1,1) LINE TO (2,1) TO (2,2) TO (1,2) TO CLOSE
  START(5,5) LINE TO (6,5) TO (6,6) TO (5,6) TO CLOSE
REGION 3  { again assumes the default }
  START(3,3) LINE TO (4,3) TO (4,4) TO (3,4) TO CLOSE
```

(2) 1 次元のリージョン

1 次元のドメインの場合、閉ループを描くことによって REGION はある有限の領域を仕切るという考え方は成り立たなくなります。1 次元の場合には始点から終点までのパスを定義するだけで十分です（CLOSE を指定すると逆向きのパスも設定されてしまうので深刻な問題が生じます）。

例えばステートメント

```
REGION 1
  START(0) LINE TO (5)
```

はこれだけで 1 次元座標系における位置 0 から 5 に至るリージョンを定義します。

2 次元、3 次元の場合との文法上の整合性を維持するため、座標値に対する括弧の省略は許されません。

REGION の持つその他の特性（後続のリージョンは先行するリージョンの上に重ねて配置され、材質の特性は REGION キーワードに続けて記述する、等）は 1 次元ドメインの場合にも有効です。

(3) 3 次元におけるリージョン

3 次元ドメインにおける REGION の概念は 2 次元に対するものと同様の特質を保持しています。

REGION は形状の 2 次元への射影における区画 (partition) であり、それを EXTRUSION の指定に従って第 3 の次元に押し出した (extrude) ものです。

3 次元における具体的な区画 (compartment) は射影上の REGION と extrusion 中の LAYER を特定することによってユニークに規定されます。

BOUNDARIES セクション中の設定をどう 3 次元に拡張するかに関する議論についてはユーザガイドをご参照ください。

(4) 3 次元におけるリージョナルパラメータ

3 次元の problem において REGION 内でパラメータの再定義を行うと、そのリージョン上のレイヤスタックに属するすべてのレイヤにおいて該当パラメータの値が再定義されることになります。再定義を特定のレイヤに限定するためには次のように LAYER 修飾子を使用してください。

```
LAYER number name = new_expression
LAYER "layer_name" name = new_expression
```

指定された LAYER 修飾子は後続のパラメータ再定義項目に対し、それらが別の LAYER 修飾子や機能的に異なるクローズによってブレークされるまで有効となります。

次はパラメータ K を種々のコンテクストで再定義した例を示しています。

Example:

```
DEFINITIONS
  K=1                {1}

BOUNDARIES
  LAYER 1 K=2        {2}
  REGION 1
    K=3              {3}
    LAYER 2 K=4      {4}
    START(0,0) LINE TO ...
```

Line {1} ではデフォルト値を設定しています。
Line {2} はすべてのリージョンを対象にレイヤ 1 中での値を規定します。
Line {3} はリージョン 1 中での値を規定するもので、全レイヤが対象になります。
Line {4} の設定はリージョン 1 中のレイヤ 2 のみが対象となります。

(5) 限定リージョン

3 次元の問題においては多くの形状がそのままでは extrusion モデルにフィットしません。特に extrude された次元においては限定的な場所にしか存在しないという形状が良くあります。そのような場合、ドメイン全体にわたって extrude するとメッシュが粗いものになってしまいます。

この問題に対処するため FlexPDE では限定リージョン (LIMITED REGION) という機能を用意しています。

LIMITED REGION は特定のレイヤ、または面上にしか存在しないリージョンとして定義されます。

LIMITED REGION は REGION 定義のボディ中で明示されたレイヤ、または面上でのみ構成されます。

この種の構造の好例としてはトランジスタを挙げることができます。デバイスの接合部はドメイン内の非常に薄い層にしか存在せず、体積の大半は基盤によって占められる形となります。

FlexPDE の旧バージョンではこの接合部の形状が extrude された次元全体に伝播されメッシュが作成されていました。しかしバージョン 4 になって構造を単一の、あるいは少数のレイヤに限定することが可能になりました。

次の例では 1 辺が 3 の立方体の中央部に薄さが 0.2 の直方体の構造が定義されています。この小さな構造はレイヤ 2 メッシュの中にのみ存在することになります。

```
EXTRUSION Z=0, 1.4, 1.6, 3

BOUNDARIES
  REGION 1
    START(0,0) LINE TO (3,0) TO (3,3) TO (3,0) TO CLOSE
  LIMITED REGION 2
    LAYER 2 K=9
    START(1.4,1.4)
    LINE TO (1.6,1.4) TO (1.6,1.6) TO (1.4,1.4) TO CLOSE
```

ユーザガイドにはグラフィックスも含めた限定リージョンに関する解説があります。

Examples:

Samples | Misc | 3D_Domains | 3D_Limited_Region.pde

(6) 空のレイヤ

3 次元の問題においては extrude されたドメイン中に穴、すなわち除去されたリージョン (excluded regions) を定義することが必要になることがあります。これは VOID 修飾子を用いることによって可能になります。VOID はパラメータ再定義の構文を持っています。

次の例では 1 辺が 3 の立方体の中央部に 1 辺が 1 の立方体の穴が設定されています。

```
EXTRUSION Z=0,1,2,3

BOUNDARIES
  REGION 1
    START(0,0) LINE TO (3,0) TO (3,3) TO (3,0) TO CLOSE
  REGION 2
    LAYER 2 VOID
    START(1,1) LINE TO (2,1) TO (2,2) TO (1,2) TO CLOSE
```

Examples:

Samples | Misc | 3D_Domains | 3D_Void.pde

4.11.4 ドメインの除去

EXCLUDE サブセクションはいくつかの REGION サブセクションの一部に重なる形で閉じたドメインを規定する場合に使用されます。EXCLUDE サブセクションによって規定されたドメインは系から除去されます。EXCLUDE サブセクションはそれが覆うことになる REGION サブセクションよりも後に記述されなくてはなりません。

EXCLUDE サブセクションは REGION サブセクションと同様に、LINE や ARC セグメントを用いて構成できます。

4.11.5 フィーチャ

FEATURE サブセクションはドメイン上で閉じた形にならないエンティティを記述する場合に使用されます。

FEATURE サブセクションは REGION サブセクションと同様に、LINE や ARC セグメントを用いて構成できます。ただし FEATURE サブセクションは予約語 CLOSE で終わることはありません。

FEATURE はノードやセルの辺によって明示的に表現されることになります。

FEATURE サブセクションは problem に線状のソースが含まれる場合、不規則なパスに沿って積分を計算したい場合、あるいはグリッドの明示的な制御が必要な場合に用いられます。

3 次元の problem において extrusion 面の傾きに鋭い変化がある場合には、その輪郭を描くために FEATURE の機能を使用してください。メッシュラインがそのような面のブレークに沿ったものでないと面のモデリングが粗いものとなってしまいます。

Example:

```
REGION 1   { an outer box }
  START(0,0) LINE TO (10,0) TO (10,10) TO (0,10) TO CLOSE

FEATURE   { with a diagonal gridding line }
  START(0,0) LINE TO (10,10)
```

4.11.6　リージョンの設定順序

厳密にこうでなくてはならないというものでもありませんが、すべての REGION サブセクションは EXCLUDE や FEATURE サブセクションに先行する形で、またすべての EXCLUDE サブセクションは FEATURE サブセクションに先行する形で設定するようにしてください。

さらに先頭の REGION サブセクションは problem のドメイン全体に対応させるべく、その外周をなぞる形で設定するようにしてください。

後の方で定義された REGION は前の方で定義されたものを上書きするものであると仮定されます。REGION にアサインされた属性についても同様のことが言えます。

4.11.7　リージョンの番号付け

REGION, EXCLUDE, FEATURE サブセクションには番号や名前をアサインすることができます。

番号をアサインするときには 1 から始まる上昇順にアサインしてください。番号は常にアサインするようにしてください。

名前をアサインする場合には引用符で囲んだ文字列を、予約語である REGION, EXCLUDE, FEATURE の直後か、それらに対する番号の直後に配置します。名前は REGION, EXCLUDE, FEATURE の中でユニークなものでなくてはなりません。

これらの名称が設定されているとリージョンに限定したプロットや体積積分を行う場合に便利です。

Example:

```
REGION 2 'Thing'
  {...}

PLOTS
  contour(u) on 'Thing'
```

4.11.8 フィレットとベベル

パス中の任意の点に続ける形で FILLET(radius)、あるいは BEVEL(length) という指定を加えることができます。その場合、該当する点は指定された半径の円弧によって、あるいは指定された長さのベベルによって置き換えられます。ただしいくつかのセグメントが交差する点に対し FILLET や BEVEL は使用しないでください。混乱が生ずる場合があります。

Example:

```
LINE TO (1,1) FILLET(0.01)
```

Examples:

```
Samples | Misc | Fillet.pde
```

4.11.9 境界条件

境界上のセグメントに対し次の形式の境界条件を設定することができます。

```
VALUE ( variable ) = Expression

NATURAL ( variable ) = Expression
LOAD ( variable ) = Expression

CONTACT (variable) = Expression

VELOCITY (variable) = Expression

NOBC ( variable )
```

境界条件の記述の中で指定されている variable により、その境界条件の適用対象である方程式が特定されることになります。

VALUE（ディリクレ）境界条件

VALUE 境界条件は、連続したいくつかの境界セグメント上で指定された変数に対する方程式の解が Expression の値に等しくなることを要請します。Expression としては定数と座標変数のみによって記述された数式の他に、値やシステム変数の導関数を含む関係式であっても構いません。

NATURAL（自然）境界条件

NATURAL 境界条件と LOAD 境界条件とは同義語です。これらは発散定理から導かれる一般化された流束 (generalized flux) に関する境界条件を表します。Expression としては定数と座標変数のみ

によって記述された数式の他に、値やシステム変数の導関数を含む関係式であっても構いません。自然境界条件はポアッソン方程式の場合にはノイマン (Neumann) 境界条件に帰着されます。自然境界条件の実装に関する情報についてはユーザガイドマニュアルをご参照ください。

CONTACT 境界条件

後述する“ジャンプ型境界”の項を参照ください。

VELOCITY 境界条件

この境界条件は境界値として指定された時間微分値を課することになります（時間依存型の問題に限る)。この条件は移動型境界を規定する場合に有用です。ある座標変数に対し適用される速度変数を宣言した場合、VELOCITY() 境界条件を用い、代替変数を境界上でのメッシュ速度変数にロックさせることになります。

現行境界条件の終結

NOBC(VARIABLE) は現行パス上においてそれまでに設定されていた境界条件を終結させる場合に使用します。それはデフォルトの境界条件である NATURAL(VARIABLE)=0 を指定したことと等価です（ただしその場合には“多重境界条件指定”という診断にはなりません)。

Note: 旧バージョンでサポートされていた NEUMANN, DNORMAL, DTANGENTIAL という境界条件は信頼性に欠けるため削除されました。将来バージョンでは復帰の可能性もありますが、多くの場合、微分型境界条件は NATURAL 境界条件によってより適切な形で適用されます。

(1) 境界条件ステートメントのシンタックス

セグメントの境界条件はそれらを BOUNDARIES セクション内に配置することによって problem descriptor に追加されます。

セグメント境界条件は予約語 LINE、または ARC の直前に配置しなくてはなりません。予約語 TO に先行することは許されません。

セグメント境界条件を方程式に適用する際にはトップダウンシステムが用いられます。それぞれのパス定義中の START ポイント指定に続く形で、変数/方程式ごとに一つのセグメント境界条件が設定されます。個々の変数/方程式ごとに境界条件を指定することが望ましいわけですが、指定がなくてもエラーにはなりません。指定がなかった場合には NATURAL(VARIABLE) = 0 という境界条件が仮定されます。

新たな境界セグメントが現れた場合、このトップダウンシステムのもとでは、新たな境界条件が設定されるまではそれ以前に設定されたセグメント境界条件が引続き適用されることになります。

推奨通り REGION 1 がドメイン全体をカバーする形で定義された場合には、通常、セグメント境界条件は REGION 1 内部のセグメントに対してのみ指定する形となります。

(2) Point **型境界条件**

POINT VALUE 境界条件は座標値の指定に続き、

```
POINT VALUE ( variable ) = expression
```

というステートメントを配置することにより設定されます。指定された値は直前の座標点に対してのみ課されることになります。

POINT LOAD 境界条件は座標値の指定に続き、

```
POINT LOAD ( variable ) = expression
```

というステートメントを配置することにより設定されます。指定されたロードが直前の座標点に対してソースとして課されることになります。

注意

これらの設定を行っても期待通りの結果が得られないことがしばしばあります。

例えば拡散方程式 $\mathrm{div}(\mathrm{grad}(u)) + s = 0$ には $u = A - Br - Cr^2$ という解が存在します。ここに r は点からの距離、A, B, C は任意定数です。重ね合わせの原理により、FlexPDE は PDE に違反することなく、得られた解に対しこのような形状を点の近傍で追加することが許されます。通常、POINT VALUE 条件は解の中に鋭いスパイクをもたらし、その点における値を指定された値にまで引き上げますが、その点以外では解の形状に変化は及びません。

POINT LOAD の場合、同じ議論は成り立ちませんが、それはスケールを持たないロードであるため、点の周囲のメッシュ密度をしばしば非常に高いものにしてしまいます。

より良いアプローチは分散型のロードを用いるか、あるいは円や矩形等の広がりを持った VALUE 境界セグメントを設定する方法です。

(3) 1 次元の場合の境界条件

境界条件は境界セグメントの全長に沿って適用されるという考え方は 2 次元、3 次元の場合には適切ですが、1 次元の場合には意味を持ちません。計算の目的自体がセグメントに沿った値を決定することにあるからです。

従って 1 次元の問題の場合には上記の Point 型境界条件の設定が常に必要になります。

Example:

```
BOUNDARIES
  REGION 1
    START(0) POINT VALUE(u)=1
    LINE TO (5) POINT LOAD(u)=4
```

(4) 3 次元の場合の境界条件

3 次元の問題においてリージョン境界にセグメント境界条件を設定すると、それは該当リージョン上にあるレイヤスタック内全レイヤの“側壁”に適用されることになります。特定レイヤの側壁に限定したい場合には次のように LAYER 修飾子を使用してください。

```
LAYER number VALUE(variable) = expression
LAYER "layer_name" VALUE(variable) = expression
```

LAYER 修飾子は別の LAYER 修飾子が現れるまで、あるいはセグメントの幾何形状に関するステートメント（LINE、または ARC）が始まるまで有効であり、その間のすべての境界条件に対し適用されます。

Extrusion 面（スライス面）自体の上での境界条件は SURFACE 修飾子をその前に付けることによって設定できます。

簡単な立方体について例を示します。EXTRUSION と BOUNDARIES セクションは次のようになります。

```
EXTRUSION z = 0,1

BOUNDARIES
  SURFACE 1 VALUE(U)=0          {1}
  REGION 1
    SURFACE 2 VALUE(U)=1        {2}
    START(0,0)
    NATURAL(U)=0                {3}
      LINE TO (1,0)
    LAYER 1 NATURAL(U)=1        {4}
      LINE TO (1,1)
    NATURAL(U)=0                {5}
      LINE TO (0,1) TO CLOSE
```

Line {1} は surface 1（$z = 0$ 平面）全面において変数 u の値として固定値 0 を指定します。
Line {2} は region 1 中の上面で変数 u の値が 1 であることを指定します。
Line {3} は立方体の側面 $y = 0$ が絶縁壁であることを指定します。
Line {4} は layer 1 の側壁 $x = 1$ 上での流束（その意味は PDE に依存）を規定します。
Line {5} により側面 $y = 1$ と $x = 0$ が絶縁壁であることが規定されます。

もちろんこの例の場合、リージョンもレイヤも一つずつしかないため、region 1 や layer 1 への制約は意味を持ちません。

(5) ジャンプ型境界

デフォルトの場合、FlexPDE はすべての変数が内部の材質境界の両側で連続であると仮定します。これはメッシュノードを境界上に配置し、それらが境界の両側に位置するセル間で共用されるようにするというアレンジの帰結と言えます。

FlexPDE にはオプションが用意されており、材質境界をクロスしたときに（従属）変数値が不連続となるケースも扱えるようになっています（機能概要についてはユーザガイドを参照ください）。

この機能は接触抵抗のようなものをモデル化する場合とか、隣接するリージョン内で変数を完全に分離する場合に用いられます。

この機能を使用する場合のキーワードは CONTACT と JUMP です。

概念的には接触抵抗のモデルがベースとなっていますが、その場合、境界（JUMP）を横切ったときの電位差は次式で与えられます。

```
V2 - V1 = R*current
```

一般的なケースで言うと、“current”の役割を担うものは一般化された流束、または自然境界条件ということになります（自然境界条件に関する解説についてはユーザガイドを参照ください）。CONTACT 境界条件は NATURAL の特殊形であるわけですが、それは流束を規定すると同時に FlexPDE に対し 2 重の値を持った境界をモデル化するよう指示することになります。

従って不連続性を指定する方法は次のようになります。

```
CONTACT(V) = (1/R)*JUMP(V)
```

“NATURAL(V)”と同様、“CONTACT(V)”は任意のセルから見たときの一般化された流束の外向き法線成分を意味します。従って“JUMP(V)”は境界上の点におけるセル内部の V の値と外部の V の値の差を意味することになります。従ってこのような境界をシェアする 2 つのセルがあった場合、それぞれ符号が異なる JUMP 値と外向き法線成分が見える形となります。この場合、流束としては双方のセルにおいて同じ数値が使用されるため、自動的に流束に対する保存則が維持されます。

内部境界において CONTACT 境界条件を指定すると、その境界に沿ってメッシュノードが 2 重に生成されると共に、それらは JUMP 境界条件ステートメントに従ってカップリングされます。

(1/R) として非常に小さな値を指定することにより、境界の両側で変数を実質的に分離することができます。

Examples:

```
Samples | Misc | Discontinuous_Variables | Thermal_Contact_Resistance.pde
Samples | Misc | Discontinuous_Variables | Contact_Resistance_Heating.pde
Samples | Misc | Discontinuous_Variables |
                Transient_Contact_Resistance_Heating.pde
```

(6) 周期型境界

FlexPDE では 2 次元と 3 次元の場合に周期 (periodic) 境界条件と反周期 (antiperiodic) 境界条件の設定が行えます。

X-Y 平面上での周期性

2 次元の問題における周期性、あるいは 3 次元での extrusion 壁における周期性は PERIODIC、または ANTIPERIODIC ステートメントによってもたらされます。

PERIODIC ステートメントは境界条件の位置に置かれますがその構文は若干異なり、またその要件とそれによってもたらされる帰結はより広範なものとなります。構文は次の通りです。

```
PERIODIC ( X_mapping, Y_mapping )
ANTIPERIODIC ( X_mapping, Y_mapping )
```

マッピングの数式は直近の境界 (immediate boundary) 上の点 (x, y) を遠隔側の境界 (remote boundary) 上の点 (x', y') にどうマッピングするかを規定するものです。それは直近境界上の各点を遠隔境界上の点にマッピングするものでなくてはなりません。セグメントの端点はセグメントの端点にマップされるものとします。またその変換は可逆でなくてはなりません。マッピングされた座標点として定数は指定しないでください。それは特異な変換 (singular transformation) をもたらすことになるからです。

周期型の境界条件ステートメントはそれまで有効だった境界条件をすべて終結させ、代りに 2 つの境界上ですべての変数値を等しく設定します。遠隔境界側で境界条件を設定することも不可能ではありませんが、通常それは適切とは言えません。

周期型の境界条件ステートメントはその次に続く LINE、または ARC パスに対してのみ有効となります。これらのパスに複数のセグメントを含めることもできますが、後続の LINE、または ARC ステートメントの出現によって周期型境界条件は終結させられます（周期型境界条件を繰返し設定した場合を除く）。

Z 次元での周期性

Extrude された次元における周期性は EXTRUSION ステートメントの前に PERIODIC、または ANTIPERIODIC 修飾子を置くことによって実現されます。次はその一例です。

```
PERIODIC EXTRUSION Z=0,1,2
```

この場合、頂部と底部の extrusion 面が類似形状であるとみなされ、値が等しく（あるいはサインを反転させた形で）設定されます。

制限事項

有限要素メッシュの各ノードは高々一つの周期イメージしか持てません。このため 2 重や 3 重の周期性は直接サポートできません。通常、周期型境界においてわずかなギャップを設けるだけで、それぞれの角はメッシュ上の他の一つの角とのみ周期的な関係を持つようにできます。

Examples:

```
Samples | Misc | Periodicity | periodic.pde
Samples | Misc | Periodicity | periodaz.pde
Samples | Misc | Periodicity | antiperiodic.pde
Samples | Misc | Periodicity | 3d_xperiodic.pde
Samples | Misc | Periodicity | 3d_zperiodic.pde
Samples | Misc | Periodicity | 3d_antiperiodic.pde
```

4.11.10　固定点

2 次元の problem ドメイン上の任意の点を特別な点として指定することができます。

```
FIXED POINT (X,Y)
```

特定された点にはメッシュノードが設定されるため、周囲におけるメッシュノードの密度に影響が及ぶこともあります。

FIXED POINT で指定された点に対しては POINT VALUE や POINT LOAD 境界条件を適用することができます。

Note:　この機能は 3 次元ではまだ十分な形で実装されていません。

4.12　Front

FRONT セクションはアダプティブな regridder で用いられる追加の判定基準を定義するために使用されます。通常 FlexPDE は PDE の近似に対する推定誤差が ERRLIM で指定された値、あるいはそのデフォルト値より小さくなるまで、計算メッシュを繰返し細分化して行きます。しかし意味のある動きがある種の伝播型フロントに限られるような場合には、そのフロントの近傍でより木目の細かなメッシュ生成を行わせたいことがあります。FRONT セクションを使うことによって関連する制御パラメータの宣言が可能になります。

FRONT セクションの形式は次の通りです。

```
FRONT ( criterion, delta )
```

criterion で指定された項が各メッシュノードにおいて評価されます。ノード上での値が 0 の周囲で (-delta/2, delta/2) の範囲を越えていた場合、セルは分割されます。

すなわち criterion が 0 の周囲 delta の範囲に収まるようグリッドの細分化が行われることになります。

Examples:

Samples | Misc | Front.pde

4.13 Resolve

RESOLVE セクションはアダプティブな regridder で用いられる追加の判定基準を定義するために使用されます。通常 FlexPDE は PDE の近似に対する推定誤差が ERRLIM で指定された値、あるいはそのデフォルト値より小さくなるまで、計算メッシュを繰返し細分化して行きます。しかし算定された量をグラフ表示する場合に、望ましいメッシュ精度に至る前にこの条件が満たされてしまう場合があります。例えばシステム変数の導関数をプロットするような場合、通常それは変数自体よりもはるかにスムーズさに欠ける傾向にあるからです。RESOLVE セクションを使うことによって細かな解像度を要する関数をいくつか宣言することが可能になります。RESOLVE セクションの形式は次の通りです。

```
RESOLVE ( spec1 ), ( spec2 ), ( spec3 ) {...}
```

ここにそれぞれの spec には“(shear_stress)”のような数式、または“(shear_stress, x^2)”のような数式と重み関数の組合せが指定できます。

最も簡潔な形式としては関心の対象である数式のみを指定する方法です。この場合、それぞれの関数に関し FlexPDE は次のような操作を行います。

- 指定された関数に関し計算メッシュ上で有限要素補間を計算します。
- 厳密な関数値からのずれを算出します。
- このずれが関数のグローバルな RMS 値と ERRLIM の積を越えた場合にはセルを分割します。

重み関数を付加した場合、重要度と重みを組み合わせた関数が規定されることになるので、解像度を要する領域に制限を加えることができます。各点でずれの大きさを評価する際、その点における

重み関数が掛け合わされます。従って重みが小さい領域では精度に関する要件がより軽いものとなります。

Examples:

Samples | Misc | Resolve.pde

4.14 Time

TIME セクションは時間依存型の problem descriptor において求解対象の時間帯を指定する場合に用いられます。指定様式には次のようなものがあります。

```
FROM time1 TO time2
FROM time1 BY increment TO time2
FROM time1 TO time2 BY increment
```

ここに

time1	開始時刻
time2	終了時刻
increment	タイムステップの初期値（オプショナル）。デフォルトは 1*e*-4*(time2-time1)。

時間依存型の problem descriptor には時間帯を規定するステートメントが含まれていなくてはなりません。時間帯を指定する方法にはいくつかありますが、最も望ましいのは TIME セクションを用いて全体の時間ドメインを指定する方法です。

実行の中断

時間帯の指定には HALT ステートメントを続けて書くことができます。

```
HALT minimum
```

この指定を行った場合には自動制御されるタイムステップ値が minimum より小さくなったところで計算が中断されます。データの矛盾、あるいはパラメータ値の不連続性によってタイムステップコントローラが混乱を来たしたような場合に有効です。

```
HALT condition
```

ここで condition としては globalmax(myvariable) $<$ 204 のような任意の関係式を記述できます。いかなるタイムステップにおいてもこの条件が成立した場合には計算は中断されます。

4.15 Monitors and Plots

MONITORS セクション（オプショナル）では求解の中間段階において表示させたいグラフィックスを一覧の形で指定します。

一方、PLOTS セクション（オプショナル）では problem やステージの完了時、あるいは選択された時刻において表示させたいグラフィックスを一覧の形で指定します。

PLOTS は MONITORS と異なり、出力結果は.PG4 レコードという形でファイル出力されるので、実行終了後であっても再表示が可能です（デバッグ目的ですがグローバルセレクタ HARDMONITOR を使用することにより、MONITORS の出力を.PG4 ファイルに書き出すことができます)。

PLOTS 関連のステートメントと MONITORS 関連のステートメントとは共通の形式と機能を持っています。

PLOTS、または MONITORS ステートメントの基本形式は次の通りです。

```
display_specification ( plot_data ) display_modifiers
```

display_specification は後述するプロットタイプのいずれかでなくてはなりません。ケースによっては複数の plot_data 引数を指定できることがあります。display_modifiers はいくつあっても構いませんが、その意味は display_specification に依存します。FlexPDE でサポートされている display_modifiers の一覧については“グラフィックディスプレイ修飾子”の項を参照ください。

推奨

MONITORS の機能を利用することによって計算の進み具合に関するフィードバックを得ることができ、また問題の解決に役立つデータを表示させることもできます。特にモデル開発の初期段階においては MONITORS の機能を積極活用してください。MONITORS の場合、.pg4 ファイルへの出力は行われないため、ディスクスペースを心配する必要もありません。モデルが正常に動作するようになったらそれらを除去したりコメントアウトしてください。

Examples:

```
Samples|Misc|Plottest.pde
```

Note: すべての用例には PLOTS や MONITORS が含まれています。

4.15.1 表示/エクスポート仕様

PLOTS や MONITORS セクションには次のタイプの表示/エクスポート仕様を指定することができます。

CDF (arg1 [,arg2,...])

指定された値を netCDF v3 の形式でエクスポートすることを要求します。出力は現行の座標系、あるいは後続の ON SURFACE 修飾子の指定に従い、2 次元あるいは 3 次元の形式となります。含まれるドメインをズームさせることもできます。FILE 修飾子が続いて指定されなかった場合には、ファイル“<problem>_<sequence>.cdf”に出力されます。ステージング型、固有値型、時間依存型の問題の場合には netCDF の慣行に従い、後続の出力が同一ファイル上にスタックされて行きます。CDF は等長の矩形グリッドを用いるため、境界線はぎざぎざになることがあります。詳細を表示する場合には ZOOM を使用してください。

CONTOUR (arg)

引数に関する 2 次元等高線図の作成を要求します。等高線は等間隔に設定されます。

CONTOUR (arg1, arg2)

arg1 と arg2 双方に関する 2 次元等高線図の作成を要求します。等高線は互いに独立に等間隔に設定されます。等高線のレベルに関する付帯する表は arg1 に関係したものであり、arg2 に対する等高線にはラベルは付きません。

ELEVATION (arg1, [arg2,...]) path

横軸にパスの値を、縦軸には引数の値を取った 2 次元のグラフ（時にラインアウト (line-out) と呼ばれます）の作成を要求します。それぞれの ELEVATION には最低 1 つの引数が指定されていなくてはなりませんが、複数の引数をコンマで区切って指定することもできます。path は FROM (X1,Y1) TO (X2,Y2) という形、もしくは ON name という形で指定された線セグメントを表します。ただし name は BOUNDARIES セクション中でパスに付けられた名標です。

GRID (arg1, arg2)

計算グリッドを示す 2 次元プロットの作成を要求します。2 つの引数はその際の座標変数を意味します。グリッドは材質の変形を示す上で有用です。(3 次元の problem の場合、2 つの引数による GRID は切断面を表すことになり、ON の指定が続かなくてはなりません。この場合のグリッドプロットは計算グリッドを正確に反映したものになるとは限りません。)

GRID (arg1, arg2, arg3)

計算グリッドを示す 3 次元プロットの作成を要求します。3 つの引数はその際の座標変数を意味します。グリッドの外周面のみがプロットされます。このプロットは SURFACE プロットの場合と同様、対話形式で回転させることができます。

SUMMARY

このプロットタイプは REPORT アイテムのみからなるテキストページを定義します。SUMMARY ページをエクスポートすることもできます。

SUMMARY ('string')

SUMMARY コマンドに対し文字列の引数が指定された場合には、それがサマリページ上にページヘッダとして表示されます。

SURFACE (arg)

引数を垂直方向に取る形で 3 次元曲面を表示します。VIEWPOINT クローズが使用されなかった場合、視点の方位角はデフォルトの 216° に、距離はサイズの 3 倍に、視点の高さは 30° に設定されます。

TABLE (arg1 [,arg2,...])

リスト形式の値をテーブル形式の ASCII データとしてエクスポートするよう要求します。出力は現行の座標系、あるいは後続の ON 修飾子の指定に従い、2 次元あるいは 3 次元の形式となります。含まれるドメインをズームさせることもできます。FILE 修飾子が続いて指定されなかった場合には、ファイル“<problem>_<sequence>.tbl”に出力されます。ステージング型、固有値型、時間依存型の問題の場合にはそれぞれのステージやモードごとに別個のファイルが生成され、その名称中にはシーケンス番号が付加されます。TABLE 出力は等長の矩形グリッドを用いるため、境界の定義は失われることがあります。詳細を表示する場合には ZOOM を使用してください。

TECPLOT (arg1 [,arg2,...])

指定された値を TecPlot システムで読める形式のファイルにエクスポートすることを要求します。出力は現行の座標系に従い 2 次元あるいは 3 次元の形式となります。メッシュ全体がエクスポートされます。FILE 修飾子が続いて指定されなかった場合には、ファイル“<problem>_<sequence>.dat”に出力されます。ステージング型、固有値型、時間依存型の問題の場合には TecPlot の慣行に従い、後続の出力が同一ファイル上にスタックされて行きます。

TecPlot は実際の三角形、あるいは四面体の計算メッシュを使用するため（ただし線形の基底関数に細分化されます)、材質境界は維持されます。

TRANSFER (arg1 [,arg2,...])

リスト形式の値と有限要素メッシュデータを FlexPDE の TRANSFER、または TRANSFERMESH 入力コマンドで読める形式のファイルにエクスポートすることを要求します。この方法による FlexPDE problem 間でのデータ転送の場合、計算の精度は完全な形で維持されるため、TABLE 機能を用いた際の矩形メッシュに伴う誤差といった問題は発生しません。ただしエクスポートされたドメインをズームさせることはできません。FILE 修飾子が続いて指定されなかった場合には、ファイル“<problem>_<sequence>.dat”に出力されます。ステージング型、固有値型、時間依存型の問題はサポートされていません。このエクスポート形式では実際の計算メッシュが使用されるため、材質境界は維持されます。計算メッシュは完全な形でエクスポートされます。

VECTOR (vector)

2 次元ベクトル場を矢印によって表示します。その際、矢印の方向と大きさは引数 vector によって規定されます。それぞれの矢印の基点は座標平面上の基準点に置かれます。

VECTOR (arg1, arg2)

2 次元ベクトル場を矢印によって表示します。その際、矢印の横方向、縦方向の成分は引数 arg1, arg2 によって与えられます。それぞれの矢印の基点は座標平面上の基準点に置かれます。

VTK (arg1 [,arg2,...])

リスト形式の値を VTK(Visualization Tool Kit) 形式のファイルにエクスポートすることを要求します。この形式のデータは VisIt 等のソフトによって表示できます。出力は現行の座標系に従い 2 次元あるいは 3 次元の形式となります。メッシュ全体がエクスポートされます。FILE 修飾子が続いて指定されなかった場合には、ファイル“<problem>_<sequence>.vtk”に出力されます。ステージング型、固有値型、時間依存型の問題の場合には一群のファイルが生成され、それぞれはシーケンス番号によって識別されます。VTK 形式では実際の三角形、あるいは四面体の計算メッシュが使用されるため、材質境界は維持されます。VTK 形式は 2 次の有限要素基底関数を直接サポートしていますが、3 次には対応していません。3 次の基底関数が用いられている場合には VTKLIN を使用してください。

VTKLIN (arg1 [,arg2,...])

VTK 形式のファイルが生成されますが、FlexPDE の計算用セルは 1 次の基底関数を用いた有限要素セルに変換されます。3 次の基底関数を用いた計算結果をエクスポートする場合、あるいは視覚化ツールが 2 次の基底関数をサポートしていない場合にはこのコマンドを使用してください。

いずれのコマンドの場合にも引数は任意の数式であって構いません。

4.15.2 グラフィックディスプレイ修飾子

表示内容は次のクローズを付加することによって修飾できます。

AREA_INTEGRATE
円柱座標系での CONTOUR, SURFACE プロットにおいて、デフォルトの $2\pi r dr dz$ という体積要素の代りに $dr dz$ を用いて積分を実行するよう指示します。

AS 'string'
ディスプレイに対するラベルを評価された数式から string に変更します。

BLACK
現在のプロットをモノクロで描画します。

BMP
BMP (pixels)
BMP (pixels, penwidth)
グラフィックエクスポートファイルの自動生成を BMP 形式で行うよう指定します。pixels は水平方向のピクセル数でデフォルト値は 1024 です。penwidth は線の太さを整数（$0, 1, 2, 3$ のいずれか）で指定します（単位は 1/1000、0 は細線を意味します）。エクスポートファイルの名称は problem の名称にプロット番号とシーケンス番号を付加したものとなります。ファイル名称の変更はできません。

DROPOUT
2.21 以前のバージョンとの互換性維持のため、EXPORT や TABLE 出力中の点の中で problem ドメインを外れるものに対し“external”というマークを施します。この修飾子は FORMAT ストリング付きの EXPORT や TABLE に対してのみ有効です。

EMF

EMF (pixels)

EMF (pixels, penwidth)

Windows 版でのみ利用できる機能で、Enhanced Metafile の出力を生成します。pixels は水平方向のピクセル数でデフォルト値は 1024 です。penwidth は線の太さを整数（0, 1, 2, 3 のいずれか）で指定します（単位は 1/1000、0 は細線を意味します)。エクスポートファイルの名称は problem の名称にプロット番号とシーケンス番号を付加したものとなります。ファイル名称の変更はできません。

(**注意**： FlexPDE は曲面プロットの場合に Y ラベルや軸ラベルに対して回転フォントを使用します。Microsoft Word はこれらの画像を読んだりサイズを変更したりできますが、その画像エディタは回転フォントを扱えないため、水平配置に戻されてしまいます。)

EPS

EPS(Encapsulated PostScript) 形式の出力を生成します。グラフィックは 10×7.5 インチのランドスケープモード様式で、その解像度は 7200×5400 です。

EXPORT

MONITOR や PLOT 出力に対応したデータをファイルとしてエクスポートするよう指示します。等長の矩形グリッドが構成され、データは FlexPDE の TABLE 機能を用いて読むのに適した形式で出力されます。グリッドの大きさは該当プロットタイプに適したプロットグリッド密度によって決定されます。この機能は PRINT 修飾子の名称を変更したものです。EXPORT されたデータの様式は FORMAT 修飾子によって制御できます。

EXPORT (n)

出力されるデータグリッドの大きさを指定した形の EXPORT コマンドです。2 次元、または 3 次元プロットの場合、グリッドの大きさはそれぞれ $(n \times n)$、または $(n \times n \times n)$ となります。

EXPORT (nx, ny)

出力されるデータグリッドの大きさを明示した形の EXPORT コマンドです。

EXPORT (nx, ny, nz)

出力されるデータグリッドの大きさを明示した形の EXPORT コマンドです。

FILE 'string'

EXPORT/PRINT 修飾子によって生成されるファイルに対する名称を string に変更します。

FIXED RANGE (arg1, arg2)
変数軸に関し動的に設定される値域を用いるのではなく、最小値として arg1 を、最大値として arg2 を用いるよう指示します。この範囲を外れたデータはプロットされません。(下記 RANGE も参照)

FORMAT 'string'
この修飾子は EXPORT/PRINT 修飾子や TABLE 出力コマンドのデフォルト様式を置き換える場合に使用します。この修飾子が使用された場合、出力はエクスポートグリッド中の各点に対し別個の行が割り当てられる形で構成されます。行の内容は次に示すようなフォーマットストリングによって完全に制御されます。

- "#"を除くすべての文字はそのまま出力行にコピーされる。
- "#"はエスケープ文字として解釈され、種々のオプションが"#"に続く文字によって選択される。
  - #x, #y, #z 及び#t はデータ点の空間座標値、時間座標値を出力する。
  - #1 から#9 はプロット関数リスト中の対応する要素の値を出力する。
  - #b はタブ制御文字を出力する。
  - #r は残りのフォーマットストリングをプロットリスト中のプロット関数ごとに繰り返すことを指示する。
  - 繰返しのストリング内での#i はプロット関数リスト中での現行要素の値を出力する。

用例については"export_format", "export_history"を参照ください。

FRAME (X, Y, Wide, High)
Problem ドメインのサイズとは無関係にプロット用の座標フレームを指定された値にセットします。メッシュ移動型の問題において一定の大きさを保ったプロットの作成が可能になります。"Samples | Moving_Mesh | Piston.pde"を参照ください。

GRAY
現行のプロットをデフォルトのカラーパレットではなく 32 階調のグレースケールで描画します。

INTEGRATE
プロットされた関数の下部に積分値を出力させます。CONTOUR, SURFACE プロットの場合、この積分は体積積分 (直交座標系の場合の体積要素は $dxdy*1$、円柱座標系の場合の体積要素は $2\pi rdrdz$) となります。一方 ELEVATION の場合、この積分は面積分 (要素は直交座標系の場合 $dl*1$、円柱座標系の場合 $2\pi rdl$) となります。(Area_Integrate, Line_integrate の項も参照)
このコマンドの場合、プロットグリッド上で積分が実行されるのに対し、REPORT(INTEGRAL(...)) の場合には計算グリッド上で積分が実行されるという違いがあります。

この修飾子をグローバルに適用させたい場合には SELECT セクション中で PLOTINTEGRATE を指定してください。

LINE_INTEGRATE
円柱座標系での ELEVATION プロットにおいて、デフォルトの $2\pi rdl$ という要素の代りに dl を用いて積分を実行するよう指示します。

LOG
LINLOG
LOGLIN
LOGLOG
スケーリングを変更します（デフォルトは線形スケール）。LOG は LINLOG と同じ意味で、データ座標に対数スケールを使用します。

<lx><ly><lz>
スケーリングを変更します（デフォルトは線形スケール）。<lx>, <ly>, <lz> のいずれも LIN か LOG のどちらかであり、対応する座標軸のスケーリングを規定します。

LOG (number)
表示する 10 進桁数を number に制限します。グローバルに設定する場合はセレクタ LOGLIMIT を指定してください。

MERGE
すべてのステージやプロット時刻に対応した EXPORT 出力を単一のファイルに併合するよう指示します（TECPLOT 出力の場合はこれがデフォルトになります）。グローバルに設定する場合には SELECT PRINTMERGE を使用してください。

MESH
SURFACE プロットにおいて、メッシュ表面の陰線描画という形で面をプロットするよう指示します。ハードコピー装置の種類によってはこの指定の方がより適切である場合があります。

NOHEADER
EXPORT 出力から problem 識別用のヘッダを除去します。

NOMERGE
各ステージやプロット時刻に対応した EXPORT 出力を別個のファイルに出力するよう指示します（これが EXPORT 出力の場合のデフォルトです）。

NOMINMAX
等高線図において最小値と最大値を示す“o”と“x”のマークを除去するよう指示します。

NORM
VECTOR プロットにおいてすべてのベクトルを等長でプロットすることを指示します。大きさの違いはカラーのみで表現されます。

NOTAGS
等高線図や elevation プロットにおいてラベリングタグを付けないよう指示します。グローバルに設定する場合には SELECT NOTAGS を指定してください。

NOTIPS
VECTOR プロットにおいて矢印を矢じりのない直線だけでプロットするよう指示します。線分の中点が基準点に置かれます。

ON "name"
ON LAYER number
ON LAYER "name"
ON REGION number
ON REGION "name"
ON SURFACE number
ON SURFACE "name"
ON equation
表示は選択されたリージョン、面、レイヤのみに限定されます。“プロットドメインの制御”の項を参照ください。

PAINTED
等高線の間をカラーで塗りつぶすよう指示します（通常の場合よりも時間を要します)。

PAINTMATERIALS
PAINTREGIONS
カラーで塗りつぶされたグリッドプロットを描画します。これらのローカルなフラグは SELECT セクション中の対応するフラグと同義であり、それらに優先します。ただし影響範囲は現行のプロットに限られます。

PNG
PNG (pixels)
PNG (pixels, penwidth)
グラフィックエクスポートファイルの自動生成を PNG 形式で行うよう指定します。pixels は水平方向のピクセル数でデフォルト値は 1024 です。penwidth は線の太さを整数（0, 1, 2, 3 のいずれか）で指定します（単位は 1/1000、0 は細線を意味します)。エクスポートファイルの名称は problem の名称にプロット番号とシーケンス番号を付加したものとなります。ファイル名称の変更はできません。

POINTS = n
デフォルトのプロットグリッドサイズを上書きし、n を使用するよう指示します。2 次元や 3 次元のエクスポートの場合、すべての次元でこの値が使用されます。

POINTS = (nx , ny)
2 次元のエクスポートの場合にグリッドサイズを明示する様式です。

POINTS = (nx, ny, nz)
3 次元のエクスポートの場合にグリッドサイズを明示する様式です。

PPM
PPM (pixels)
PPM (pixels, penwidth)
グラフィックエクスポートファイルの自動生成を PPM 形式で行うよう指定します。pixels は水平方向のピクセル数でデフォルト値は 1024 です。penwidth は線の太さを整数（0, 1, 2, 3 のいずれか）で指定します（単位は 1/1000、0 は細線を意味します)。エクスポートファイルの名称は problem の名称にプロット番号とシーケンス番号を付加したものとなります。ファイル名称の変更はできません。

PRINT
EXPORT と同一機能です。

PRINT (n)
EXPORT(n) と同一機能です。

PRINT (nx, ny)
EXPORT(nx,ny) と同一機能です。

PRINT (nx, ny, nz)
EXPORT(nx,ny,nz) と同一機能です。

RANGE (arg1, arg2)
(従属) 変数軸に対し動的にセットされる値域に代り、最小値を arg1 に、最大値を arg2 に設定します。ただし計算された変数値が指定された範囲を越えた場合、指定値は無視され、動的に計算された値が用いられます。(FIXED RANGE の項も参照)

VIEWPOINT(X, Y, angle)
SURFACE プロットに対し、視点に関する方位角、遠近距離、仰角を設定します。

VOL_INTEGRATE
円柱座標系での CONTOUR, SURFACE プロットにおいて、$2\pi rdrdz$ という体積要素を用いて積分を行うよう指示します。これはデフォルトであり INTEGRATE と等価です。(INTEGRATE, AREA_INTEGRATE の項も参照)

XPM
XPM (pixels)
XPM (pixels, penwidth)
グラフィックエクスポートファイルの自動生成を XPM 形式で行うよう指定します。pixels は水平方向のピクセル数でデフォルト値は 1024 です。penwidth は線の太さを整数 ($0, 1, 2, 3$ のいずれか) で指定します (単位は 1/1000、0 は細線を意味します)。エクスポートファイルの名称は problem の名称にプロット番号とシーケンス番号を付加したものとなります。ファイル名称の変更はできません。

ZOOM (X, Y, Wide, High)
表示上、またはエクスポート上で選択された領域を拡大します。その場合、(X, Y) は領域の左下隅を、(Wide, High) は拡大対象の領域の広がりを規定します。3 次元の切断面の場合、X, Y という座標変数は切断面上での水平軸、垂直軸を意味します。

ZOOM (X, Y, Z, Xsize, Ysize, Zsize)
エクスポート上で選択された 3 次元領域を拡大します。その場合、(X, Y, Z) は領域の左下隅を、(Xsize, Ysize, Zsize) は拡大対象の領域の広がりを規定します。

4.15.3　プロットドメインの制御

ON セレクタ

プロットデータを作成するドメインを制御する基本的なメカニズムは ON ステートメントで種々の形式があります。

```
ON "name"
ON REGION "name"
ON REGION number
```

また 3 次元の problem においては次のような形式も使用できます。

```
ON LAYER "name"
ON SURFACE "name"
ON LAYER number
ON SURFACE number
ON equation
```

1 番目の形式の場合、"name"の定義内容によって境界パス、リージョン、レイヤ、面が選択されることになります。（実際、SURFACE "name"等の指定方法は冗長と言えます。なぜなら面が指定されたことは"name"自体からわかるからです。しかし冗長な形式でも問題なく使用できます。）

多くの場合、特に 3 次元の場合、一つのプロットを規定するのに複数の ON クローズが用いられることがあります。なぜならそれぞれの ON クローズが既に設定されていた内容にさらに条件を課す形になるからです。プロットステートメントの ON クローズと INTEGRAL 演算子の引数との間には直接的な対応関係があります。しかし積分の場合に指定できるオプションの中にはプロットオプションに対応しないものもあります。

2 次元の場合、何ら制約のない面積プロットは problem ドメイン全体を対象に計算されます。

等高線図、面プロット、グリッドプロット、ベクトルプロット

等高線図、面プロット（3 次元曲面表示）、グリッドプロット、ベクトルプロットは何らかの 2 次元データ面上に構成されます。このため 3 次元の problem において、これらのコマンドを ON クローズを伴わない形で発行したとすればそれは不完全なものとなります。ON クローズで指定するのは extrusion 面の名称のこともあれば切断面を規定する方程式のこともあります（射影面の境界パスを使うことはできません）。3 次元の場合の用例をいくつか示します。

`CONTOUR(...) ON SURFACE 2`
第 2 の extrusion 面上で評価されるデータをもとに等高線図を作成します。

`CONTOUR(...) ON SURFACE "top"`
`"top"`という名称の extrusion 面上で評価されるデータをもとに等高線図を作成します。

`CONTOUR(...) ON X=Y`
$x = y$ で表される切断面上で評価されるデータをもとに等高線図を作成します。

データ面の基本的な定義に加え、`ON` クローズは任意のリージョンやレイヤに表示範囲を限定する場合にも使用されます。2 次元の場合、`REGION` という制限項を加えるとドメイン上の該当リージョン部分のみが表示されることになります。

`CONTOUR(...) ON REGION 2`
`REGION 2` 上で評価されるデータをもとに等高線図を作成します。

次は 3 次元の場合の例です。

`CONTOUR(...) ON SURFACE 2 ON REGION 2`
Extrusion 面である `SURFACE 2` 上で、かつベース平面上の `REGION 2` を投影した部分のみを対象に、その上で評価されるデータに基づき等高線図を作成します。

`CONTOUR(...) ON SURFACE 2 ON REGION 2 ON LAYER 3`
Extrusion 面である `SURFACE 2` 上の `REGION 2` 部分、及び `LAYER 3` で評価されるデータに基づき等高線図を作成します。

3 次元における切断面

等高線図、面プロット、ベクトルプロットは切断面上でも作成できます。その際、切断面は方程式で規定します。

`CONTOUR(...) ON X = expression`
x が指定された値を取る y-z 平面上でデータを評価し等高線図を作成します。

切断面はこのような座標平面に限りません。

`CONTOUR(...) ON X=Y`
平面 $x = y$（x-y 平面上の直線 $y = x$ と z 軸を含む平面）上でデータを評価し等高線図を作成します。

斜めの切断面上に表示される座標系の原点はドメイン座標系の原点に最も近い点となります。また座標軸はドメイン座標系の最も近い座標軸に沿ったものとなるよう選択されます。

Elevation プロット

Elevation プロットは直線の両端を指定する形で作成できます。

```
ELEVATION(...)  FROM (x1,y1) TO (x2,y2)
ELEVATION(...)  FROM (x1,y1,z1) TO (x2,y2,z2)
```

プロットは指定された点の間を結ぶ直線上に表示されます。これらの端点は problem ドメイン内に収まるケースもあれば、それをはみ出して設定される場合もあります。その場合、はみ出した部分のプロットラインは切り捨てられます。

2 次元の場合に限り、elevation プロットは境界パスの名称によっても設定できます。

```
ELEVATION(...)  ON "outer_boundary"
```

このような境界パス上の elevation の場合、それをさらに特定のリージョンに限定することもできます。

```
ELEVATION(...)  ON "inner_boundary" ON REGION "core"
```

この形式の場合、プロット関数の評価は"core"という名前のリージョン内で行われます。ただし"core"は境界線"inner_boundary"に隣接するリージョンの一つであると仮定されます。

ベクトル成分の符号

境界パス上の elevation としてはベクトルの法線成分や接線成分が用いられることが良くあります。そのような場合、方向の意味は自然境界条件の場合の意味と同一となります。

- 正の法線は評価対象の領域から外向きの方向を意味します。
- 正の接線は評価対象の領域から見て反時計方向を意味します。

3 次元の extrusion 面上でのベクトルの法線成分についても同様です。

- 正の法線は評価対象の領域から外向きの方向を意味します。

4.15.4 レポート

グラフィック表示に関する指定の後に以下のようなクローズを追加することにより、プロットページに物理量に関するレポートを出力させることができます。

`REPORT expression`
ディスプレイの下部に“数式 = 値”という形式のテキスト行を追加します。ここに expression は数式であり、積分も含めることができます。REPORT クローズは複数あっても構いません。REPORT は境界上の積分値や面積分、及びそれらの関数値をレポートするのに便利です。

`REPORT expression AS 'string'`
ラベル指定の REPORT で、“string = 値”というテキストが出力されます。

`REPORT('string')`
`REPORT 'string'`
REPORT の並びの中に'string' を挿入します。

4.15.5 ウィンドウ構成

複数の MONITORS や PLOTS 出力を要求した場合、FlexPDE はそれらを別個のウィンドウ上に作成の上サイズを調整し、すべてのウィンドウをタイル張りの形で画面上に表示します。個々のウィンドウを個別に拡大したりアイコン化することはできません。どのウィンドウもそれをダブルクリックすることで最大化できます（右クリックの上、メニューから操作する形でも最大化できます)。

定常状態型と固有値型の問題の場合、MONITORS は求解中に表示され、計算終了後 PLOTS により置き換えられます。時間依存型問題の場合、MONITORS, PLOTS, HISTORIES は常時表示されます。

4.15.6 Monitors 出力 - 定常状態型

定常状態型の問題の場合、リストされた MONITORS は再グリッドのたびに表示されます。それに加えて Newton-Raphson 反復の都度（非線形問題)、あるいは残差反復の都度（線形問題)、途中結果を示すモニタ出力が表示されます（ただし、前回のモニタ出力から十分な時間が経過している場合)。

4.15.7　Monitors/Plots 出力 - 時間依存型

時間依存型問題の場合、ディスプレイの指定の前には表示時刻に関するステートメントがなくてはなりません。この表示時刻に関するステートメントは次のいずれかの形式を取ります。

```
FOR CYCLE = number
```

この場合には指定された数のタイムステップごとに表示内容が更新されます。

```
FOR T = timeset1 [ timeset2 ...  ]
```

ここで個々の timeset は特定の時刻か、または

```
t1 BY deltat TO t2
```

という形で指定された一群の値を意味します。この指定の場合、timeset で指定された時刻ごとに表示内容が更新されます。

この表示時刻に関する宣言の後に任意数のプロットコマンドを記述することができます。その場合、時刻設定は後続のプロットコマンドすべてに適用されます。プロットコマンドごとに時刻の宣言を行う必要はありません。

表示時刻に関する宣言を複数設定することもできます。その場合、それぞれの表示時刻に関する宣言は次の宣言が現れるまでの範囲で、あるいは MONITORS や PLOTS セクションが終了するまでの範囲で、それに続くすべてのプロットコマンドに適用されます。

Examples:

Samples | Time_Dependent | Heatflow | Float_Zone.pde
Samples | Time_Dependent フォルダ中のその他の problems

4.15.8　ハードコピー

任意のプロットウィンドウを右クリックすると、それが一覧形式（タイル張り）か最大化されているかによらずメニューが表示されるので、印刷したりエクスポートすることができます（プロットによっては回転操作も行えます）。

プロットされた値をテキスト形式でディスクに出力したい場合には、descriptor 上のプロットコマンドで EXPORT 修飾子を使い操作してください。

4.15.9 グラフィックスのエクスポート

ビットマップ

表示されたプロットウィンドウを右クリックするとメニューが表示されるのでその中から“Export”を選択してください。ビットマップ形式での画像エクスポートを支援するためのダイアログが表示されます。現在選択できるオプションには BMP, PNG, PPM, XPM があります。詳細は操作ガイドを参照ください。
プロットコマンド修飾子を用いるとこの画像形式の選択を自動化させることができます。

保存されたグラフィックス

PLOTS セクション中のすべてのグラフィックスは拡張子が“.PG5”のディスクファイルに圧縮された形式で保存されます。
これらのファイルは別途“File”メニューから“View”と操作することによって再表示させることができます。システムによってはファイルマネジャ上で“.PG5”ファイルをダブルクリックするだけで再表示が行えます。詳細は操作ガイドを参照ください。

画面キャプチャ

任意のグラフィックスは JASC(www.jasc.com) の PaintShopPro で提供されているような画面キャプチャの機能を利用することにより、他のウィンドウベースのプログラム中に paste することができます。

ファイルのエクスポート

プロットタイプ CDF, TABLE, TECPLOT, VTK を選択した場合には他のアプリケーション用にデータをエクスポートすることができます。TRANSFER を用いた場合には FlexPDE 間でのデータ転送が行えます。詳細についてはセクション 4.15.1 を参照ください。

Examples:

```
Samples|Misc|Printest.pde
Samples|Misc|Import-Export|Export.pde
Samples|Misc|Import-Export|Export_Format.pde
Samples|Misc|Import-Export|Export_History.pde
```

4.15.10 用例

PLOTS、MONITORS の用例については Samples | Misc | Plottest.pde 中のサンプルを参照ください。
プロットデータのエクスポートについては Samples | Misc | Printest.pde 中のサンプルを参照ください。
表示を伴わないエクスポートの用例については Samples | Misc | Import-Export | Export.pde 中のサンプルを参照ください。

4.16 Histories

HISTORIES セクション（オプショナル）ではヒストリの欲しい値を指定します。複数の HISTORY ステートメントを並べることができますが、それらはすべて次の形式でなくてはなりません。

```
HISTORY ( arg1 [ ,arg2,...]  )
HISTORY ( arg1 [ ,arg2,...]  )  AT (X1,Y1) [ (X2,Y2)...]
```

座標値は problem 上でヒストリを記録したい位置を表します。座標値なしの場合には arg はスカラー値として評価されなくてはなりません。

PLOTS と MONITORS で使用できる修飾子とレポートは HISTORY ステートメントにも適用できます。

HISTORIES の表示は AUTOHIST セレクタによって制御されます（デフォルトは ON）。デフォルトの設定の場合、すべての HISTORY は MONITORS か PLOTS の更新に伴い、自動的に更新され表示されます。

HISTORY ステートメントを直接 MONITORS セクションや PLOTS セクション内に配置することも可能です。

ステージングとヒストリ

HISTORY ステートメントは STAGED problem や時間依存型 problem 内でも使用できます。その場合、デフォルトの横軸にはステージ番号が取られます。横軸用に別の値を用いたい場合には

```
VERSUS expression
```

というクローズを追加してください。この場合、種々のステージにおける expression の値がプロット軸として使用されます。

4.17 End

すべての problem descriptor には `END` セクションがなくてはなりません。予約語 `END` よりも後ろにあるステートメントはすべて無視されコメントとして扱われることになります。従って problem に関する種々の注釈は `END` の後に記述することができます。

第 5 章

バッチ処理

一群の problem をバッチモードで実行させるためには特別な形式の descriptor を使用します。

BATCH という語で始まる単一のセクションにより descriptor がバッチ制御ファイルであると認識されます。このヘッダに続く形で一連の名称をそれぞれ引用符で囲んで入力します。それぞれを区切るためにコンマを使用しても構いません。各行に置ける名前の数に特に制限はありません。これらの名称は実行すべき problem descriptor の名称です。名称にはディレクトリパスが含まれていても構いません。それらはバッチ descriptor の存在するディレクトリ内の相対パスを表すものと仮定されます。拡張子".pde"の指定は必要ありません。この一覧は END ステートメントで終結させます。

一例を示すと次のようになります。

```
BATCH
{ use the correct separators for your operating system }

  "misc\table", "steady_state\heat_flow\slider"
  "steady_state\stress\3d_bimetal"

END
```

Problem リスト全体が即座にチェックされ、構文エラーがあった場合には報告されます。リスト中のすべてのファイルの位置が確認され、見つからないファイルがあった場合にはその旨報告されます。

個々の problem は順番に最後まで実行されます。実行時のステータス情報はバッチ descriptor を含むディレクトリ内にログファイルとして出力されます。このファイルには拡張子".log"が付けられますが、そのファイル名はバッチ descriptor の名称と同一です。すべての problem に関するステータス情報がこの一つのファイルに集約して出力されます。一方、個々の problem からのグ

ラフィックス出力は該当 descriptor の置かれているディレクトリ中に.PG4 ファイルとして出力されます。実行終了後 FlexPDE を再起動し、VIEW メニューアイテムを使うことによってこれらのグラフィックス出力を表示させることができます。

簡単な名称は引用符なしにも入力できますが、中にスペースやパスセパレータ、予約語等が含まれていた場合にはエラーとなります。

第 6 章

コマンドによる実行

FlexPDE をコマンドラインから実行させる場合、あるいは他のアプリケーションからのサブタスクとして実行させる場合、その実行を制御するためのコマンドラインスイッチがいくつか用意されています。

-R	コマンドラインで指定されたファイルの実行。編集モードには入らない。
-V	コマンドラインで指定されたファイルの表示。編集モードには入らない。
-X	Problem の実行が完了したとき FlexPDE を終了させる。
-M	最小化されたモードで（アイコンとして）実行。
-Q	静かに（“quietly”）実行。-R -X -M の組合せ。
-S	静かに（“silently”）実行。エラーレポートを抑止した-Q。

索引