AMPL：数理計画のためのモデリング言語

第 9 章

データの指定

集合とパラメータの値をどう表すか — data コマンドとデータ文の書式

モデルとデータの分離 ｜ リスト形式 ｜ 表形式 ｜ read コマンド

本章の内容

Agenda

9.1 整形されたデータ：data コマンド
モデルモードとデータモードの切替、構文ルール、複数ファイル

9.2 リスト形式のデータ
一次元〜多次元の集合・パラメータ、テンプレートによるスライス

9.3 表形式のデータ
二次元の表、転置・分割、高次元データのスライス表・多次元表

9.4 データ文のその他の機能
既定値（default）、添字付き集合の集まり、変数の初期値

9.5 整形されていないデータの読み込み：read コマンド
item-list による柔軟な値の取り込み

モデルモードとデータモードの切替

9.1

- param・var の宣言や solve・display コマンドは「モデルモード」で実行される（作業の大半で標準）
- モデルモードは集合・パラメータの長い値のリストを読むには不便
- data キーワードで開始すると「データモード」に入り、データ文を読む
- data diet.dat; のようにファイル名を付ければそのファイルから読む
- ファイル名なしの data; は以降の入力をデータ文とみなす（対話的に使える）
- set/param で始まらない文（display など）や model コマンドでモデルモードに復帰

対話セッション例

```
ampl: model dietu.mod;
ampl: data;
ampl data: set MINREQ := A B1 B2 C CAL;
ampl data: set MAXREQ := A NA CAL;
ampl data: display NUTR;
set NUTR := A B1 B2 C CAL NA;
ampl:

ampl: data diet.dat;
  # ファイル名を指定して読む場合
```

データモードの構文ルール

9.1

- 空白・タブ・改行はまとめて 1 個のスペース、区切りのコンマも無視される → 読みやすい表に整形できる
- 文字列は通常、引用符で囲む必要はない
- 英字・数字・下線・ピリオド・+・- 以外の文字を含む文字列は引用符が必須（例：'A&P'）
- 引用符自体を含む文字列は種類を変えるか二重にする
- 数値に見える文字列（"+1"、"3e4"）も数値と区別するため引用符が必要

引用符の例

- "DOMINICK'S"（二重引用符で単一引用符を含める）
- 'DOMINICK''S'（引用符を内側で二重化）

同じ内部表現の数値は同じ集合要素

```
2  =  2.00  =  2.e0  =  0.02E+2
```

いずれも同一の値として扱われる

複数データファイルと重複の検査

9.1

- 複数の data コマンドを使えば、モデルの構成要素に任意の数のデータファイルから値を割り当てられる
- AMPL はどの構成要素にも値が 2 回以上割り当てられていないかを常に検査する
- 重複する割当はエラーとして指摘される（下の例）
- シナリオを変えて一部データだけ更新したい場合は、reset data や update data コマンドを先に与えれば新しい値を受け入れさせられる（第11.3節）
- let コマンドで集合・パラメータの値を直接割り当てる方法もある（第11.3節）

重複エラーの例

```
duplicate member coils
  for set PROD
context: set PROD :=
  bands coils plate coils
  >>> ; <<<
```

ポイント

ファイル数によらず「重複割当なし」が保証される。意図的な再割当には reset data / update data / let を使う。

一次元の集合とパラメータのリスト

9.2

モデル + データ

```
set PROD;
param rate {PROD} > 0;

set PROD := bands coils plate ;

param rate :=
  bands   200
  coils   140
  plate   160 ;

# 余分な空白や改行は無視される
```

- 集合の指定は要素の列挙のみ。パラメータは各要素の後に値を続けるだけ
- 空白・改行は無視されるため、表のように整形して読みやすくできる
- 集合の要素はすべて異なる必要があり、重複は AMPL が警告する
- 同様にパラメータへの二重の値割当てもエラーになる
- 要素の空リスト（:= の直後にセミコロン）で集合を空と指定できる
- ordered / circular 属性の集合は、要素の順序がデータ文の順で決まる

二次元集合：組のリストとテンプレート

9.2

組のリスト（LINKS）

```
set ORIG;  set DEST;
set LINKS within {ORIG,DEST};

set LINKS :=
  (GARY,DET) (GARY,LAN)
  (CLEV,FRA) (PITT,FRA) ... ;
```

テンプレートでスライス

```
set LINKS :=
  (GARY,*) DET LAN STL LAF
  (CLEV,*) FRA DET LAN WIN ...
  (PITT,*) FRA WIN STL FRE ;
```

データテンプレートとスライス

- 対に似ているが一方の成分が * に置き換わった式を「データテンプレート」と呼ぶ
- (GARY,*) の後にリストが続き、各項目が * に代入されて対が生成される
- こうしてできた対の集まりが、その次元における集合の「スライス」をなす
- 第 1 成分固定なら (GARY,*)、第 2 成分固定なら (*,FRA) のように使う
- 対の中の要素順には意味がある：第 1 要素は ORIG、第 2 要素は DEST から
- 対そのものは任意の順序で書いてよい（* を含まない対は自分自身のみを指定）

二次元パラメータのリスト

9.2

- 集合 LINKS で添字づけられたパラメータ cost も、集合と同じ形式にパラメータ値を加えて指定する
- スライスのリストも同様に拡張され、暗黙の各集合要素の後にパラメータ値を置く
- パラメータのテンプレートは角括弧 [ ] で囲む（集合の丸括弧 ( ) と区別するため）
- [GARY,*] の後の DET 14 は、cost["GARY","DET"] に 14 を与えることを意味する
- 対のリストは、既定テンプレート [*,*] を使った特別な場合とみなせる

```
param cost {LINKS} >= 0;

param cost :=
  [GARY,*] DET 14  LAN 11
           STL 16  LAF 8
  [CLEV,*] FRA 27  DET  9
           LAN 12  WIN 9
           STL 26  LAF 17
  [PITT,*] FRA 24  WIN 13
           STL 28  FRE 99 ;
```

三次元以上への拡張と一般規則

9.2

1次元スライス（ROUTES）

```
set ROUTES within {ORIG,DEST,PROD};
set ROUTES :=
  (CLEV,*,bands) FRA DET LAN ...
  (GARY,*,coils) LAN STL LAF ;
```

2次元スライス（* が2つ）

```
set ROUTES :=
  (*,FRA,*) CLEV bands
            CLEV coils
            PITT bands ;
# * の数だけ後続の要素が対で続く
```

一般規則（n 次元テンプレート）

- n 次元の集合・パラメータのテンプレートは n 個の項目（正しい集合要素または *）を持つ
- 集合のテンプレートは組と同様に丸括弧、パラメータのテンプレートは角括弧で囲む
- テンプレートの後には、* 一つにつき一つの集合要素（パラメータなら値も）が続く
- * の数が異なるテンプレートを同じデータ文で混在させてもよい
- テンプレートがない箇所は、すべて * のテンプレートが仮定される

集合とパラメータを結合したリスト

9.2

冗長な書き方

```
set PROD := bands coils plate ;
param rate :=
  bands 200 coils 140 plate 160 ;
```

集合名を param 文に含めて重複を回避

```
param: PROD: rate :=
  bands 200  coils 140  plate 160 ;
```

複数パラメータをまとめる

```
param rate   := bands 200 coils 140 plate 160 ;
param profit := bands  25 coils  30 plate  29 ;
param market := bands 6000 coils 4000 plate 3500 ;

# ↓ 一つの文にまとめられる
param: rate profit market :=
  bands 200 25 6000
  coils 140 30 4000
  plate 160 29 3500 ;
```

集合＋複数パラメータも一括可能

```
param: PROD: rate profit market := ... ;
```

欠損値のドット（.）表記

9.2

- n_min は MINREQ、n_max は MAXREQ という重なりうる異なる集合で添字づけられる
- MINREQ にない NA については n_min[NA] は定義されない → ドット（.）を明示的に置く
- 空欄のままにはできない：空白は無視されるため次の値と誤って結び付く
- ドットの代わりに 0 を置くのも誤り。未定義のまま参照するとエラーになる

アクセス時のエラー例

```
error processing param n_min:
  invalid subscript n_min['NA'] discarded.
```

栄養素	n_min	n_max
A	700	20000
C	700	.
B1	0	.
B2	0	.
NA	.	50000
CAL	16000	24000

表形式データの基本

9.3

- 行のラベルが第1添字、列のラベルが第2添字を与える
- パラメータ名の後にコロン、列ラベルの後に := を置くと AMPL が表と認識する
- cost["GARY","FRA"] は 39 に設定される

cost	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	39	14	11	14	16	82	8
CLEV	27	9	12	9	26	95	17
PITT	24	14	17	13	28	99	20

図 *3-1b*：*param cost: FRA DET LAN WIN STL FRE LAF := … ;*

転置：(tr)

添字集合の一方だけが大きい場合、小さい方の集合を列に置くため表を転置できる。

```
param cost (tr):
  GARY CLEV PITT :=
  FRA   39   27   24
  DET   14    9   14
  ... ;
```

(tr) は列ラベルが第*1*添字、行ラベルが第*2*添字を表す

大きな表の分割と集合の +/- 表

9.3

- 改行は無視されるため、各行を複数行に折り返して幅を抑えられる
- 表を列方向にいくつかの部分表へ分割してもよい。新しい部分表の先頭はコロンで示す
- LINKS のような疎な添字集合では、表中にドット（.）を置いて「値なし」を示す
- 集合自体も同じ形の表で指定でき、要素なら +、非要素なら - を置く

列方向の分割（例）

```
param cost: FRA DET LAN WIN :=
   GARY  39  14  11  14
   CLEV  27   9  12   9
        : STL FRE LAF :=
   GARY  16  82   8
   CLEV  26  95  17 ;
```

LINKS	FRA	DET	LAN
GARY	-	+	+
CLEV	+	+	+
PITT	+	-	-

+ は集合の要素である対、- は要素でない対を示す（パラメータの表形式はそのまま集合にも使える）

高次元データの表：スライスと多次元表

9.3

2次元スライスの並び

```
param cost :=
 [*,*,bands]: FRA DET LAN WIN STL FRE LAF :=
  CLEV  27  9  12  .  26  .  17
  PITT  24  .  .  13  28  99  .
 [*,*,coils]: FRA DET LAN WIN STL FRE LAF :=
  GARY   .  .  11  .  16  .  8
  ... ;
```

テンプレートで配置を精密指定

```
[*,:,*] : FRA DET LAN ... :=
 CLEV bands  27   9  12 ...
# * は左（行）、: は上（列）に来る添字
```

多次元表の考え方

- 2次元を超えるデータは、二次元の「スライス」表の並びとして指定できる（テンプレートに * が2つ）
- 左右の見出しに2つ以上の添字を置き、単一の表で多次元データを表すこともできる
- 見出し行それぞれの前にコロンを置き、:= は最後の見出し行の後にのみ置く
- 既定では左→上の順に添字が取られる。(tr) を付けると上→左の順に転置される
- 表現の選び方はデータの意味に影響しない → 読みやすさ・簡潔さで自由に選べる

既定値（default 句）

- 値の与え忘れ・与えすぎはどちらもエラーになる（solve 時などに検出）
- 同じ値が何度も現れるなら default 句で繰り返し指定を避けられる
- 添字づけられた集まり全体に一様な値を与える場合にも便利（例：供給量をすべて 1 に）
- default はモデルの param 宣言中にも書ける。既定値を他のデータに依存させたい場合はこちらを使う

データ文での default

```
param cost default 9999 :=
 [*,*,bands]: FRA DET LAN ... :=
  CLEV 27 9 12 . 26 . 17
  PITT 24 . . 13 28 99 .
 [*,*,coils]: ... ;
```

24

個の 9999 が自動的に割り当てられる

```
param supply default 1;
param demand default 1;
```

割当問題：供給・需要を一律 1 に

添字付き集合の集まり／変数の初期値

9.4

集合の添字づけられた集まり

```
set PROD;
set AREA {PROD};
set AREA[bands] := east north ;
set AREA[coils] := east west export ;
```

- 集合名にキーワード set の後で添字を付ける以外、通常の集合データ文と同じ形式が使える
- 空にしたい要素は空リストで指定。データを与えない要素は、モデル側で default {} を宣言すれば警告を避けられる

変数の初期値

```
var Trans:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
  GARY  100  100  800  100  100  500  200
  CLEV  900  100  100  500  500  200  200
  PITT  100  900  100  500  100  900  200 ;
```

- param データ文と同じ形式で、パラメータ名の代わりに変数名・制約名を使えばよい（先頭を var にしてもよい）
- 初期値はソルバーへの出発点になる。非線形最適化で特に有用（第18章）

read コマンド：整形されていないデータ

9.5

- ファイルが集合名・パラメータ名・:= を含まず、値だけが順序正しく並んでいる場合に便利
- 基本形：read item-list < filename ;　値は現れる順にリストの項目へ割り当てられる
- item は「パラメータ」「{添字式} パラメータ」「{添字式} (item-list)」のいずれか（入れ子も可）
- ファイル名を省くと現在の入力（対話プロンプトなど）から読む。"-" は標準入力を表す

順序に関する注意

文字列で添字づけられた集合は、要素の内部順序で読まれる。順序を確実にするには ordered / ordered by を使うか、ファイル中に添字自体を明示する。

基本例：週次データ

```
param T > 0;
param avail {1..T} >= 0;

read T, {t in 1..T} avail[t]
  <week_data.txt;

# week_data.txt:  4  40 40 32 40
```

入れ子の item-list

```
read {p in PROD}
  (prodcost[p], invcost[p],
   {t in 1..T} (revenue[p,t],
                market[p,t]))
  <cost_data;
```

✓ まとめ

1. **data コマンドでモデルモードとデータモードを切替**
   宣言（モデル）と値（データ）を明確に分離して記述できる

2. **リスト形式はテンプレート（*）でスライスを簡潔に表現**
   1次元〜多次元の集合・パラメータに一貫した構文を適用

3. **表形式は2次元以上の添字を読みやすく配置**
   転置 (tr)、部分表への分割、+/- 記法、多次元見出しが使える

4. **default・添字付き集合の集まり・変数初期値など補助機能が豊富**
   繰り返しの排除や、ソルバーへの出発点指定に活用できる

5. **read コマンドで整形されていない外部データも柔軟に取り込める**
   item-list の入れ子で、ファイルの並びに合わせた読み込みが可能