AMPL：数理計画のためのモデリング言語

第11章

モデリングコマンド

model・*solve* から *fix*・*drop*・*relax_integrality* まで ―
解いて、直して、また解く。対話的セッションを操るコマンド群

>_

全22章シリーズ　第11回

AGENDA

本章の内容

11.1 コマンドとオプションの一般原則

対話セッションの基本構造、option コマンドによる設定

11.2 モデルとデータの設定と求解

model / data / solve コマンド、目的関数の選択、ソルバー選択

11.3 データの変更

reset data / update data、再標本化、let コマンド、丸め関数

11.4 モデルの変更

delete / purge / redeclare、fix / unfix / drop / restore、整数性の緩和

コマンド環境の一般原則

▸ 対話的セッション

ampl を起動すると ampl: プロンプトが現れ、入力されたコマンドを解釈・実行する

; セミコロンで終端

コマンドはセミコロンで終わる。1行に複数コマンドを書いても、複数行にまたがってもよい

? 継続プロンプト

コマンドの途中で改行すると ampl? が現れ、続きの入力を促す

F ファイル名

セミコロンと引用符を除く印字可能文字の並び。規則は OS が決める

Q end / quit

セッションを終えるにはこのいずれかを打ち込む

amplsession

```
ampl: display {i in ORIG, j in DEST}
ampl? sum {p in PROD} Trans[i,j,p];

# ampl?  は継続入力を示す
```

コマンドはスクリプト（第13章）にも登場する。GUIのメニュー操作も裏ではこのコマンド体系を呼び出している。

option コマンド

- **各オプションは名前と値（数値または文字列）を持つ**
  - 例：prompt1・prompt2（文字列）、display_width（数値）
- **option 名前; で現在の値を表示**
- **option 名前 値; で値を再設定**
- **名前に * を使うとワイルドカード一致**
  - option だけ、または option *; で全オプション一覧
- **reset options; で既定値に一括復元**
  - AMPLは正しいオプション名の一覧を持たない — 打ち間違いは新しいオプションとして静かに定義されるので注意

オプションを見る

```
ampl: option prompt1, display_width;
option prompt1 'ampl: ';
option display_width 79;
```

オプションを設定

```
ampl: option prompt1 "A> ", display_width 60;
A> option display_width;
option display_width 60;
```

モデルとデータの設定と求解

diet.mod を解く

```
ampl: model diet.mod; data diet.dat; solve;
MINOS 5.5: optimal solution found.
6 iterations, objective 88.2
```

- model：モデル宣言（第5〜8章）を含むファイルを名指しし、現在のモデルに追加
- data：データ値（第9章）を含むファイルを名指しし、既存の構成要素に値を与える
- solve：最適化問題をソルバーへ送り、結果を取り出す
- 複数回の model / data コマンドで、異なるファイルから積み上げられる

M モデル / データ モード

- AMPL は常に「モデルモード」で開始
- ファイル名なしの data; でデータモードへ
- ファイル名なしの model; でモデルモードへ戻る
- option・solve・subject to 等もモデルモードへ復帰させる

Σ objective コマンド

単一の目的関数を選ぶ

```
objective Total_Number;
objective Total_Cost["A&P"];
```

solve コマンドの流れ

1

問題の生成

モデルとデータから具体的な最適化問題を作る。データ不足や制限違反はここでエラー

2

前処理（プリソルブ）

問題を解きやすく縮小。実行不可能を検出しソルバー送信を省くこともある

3

ソルバーへ送信

既定ソルバーは option solver で確認・変更できる

前処理が実行不可能を検出

```
ampl: model steel4.mod;
ampl: data steel4.dat;
ampl: let avail['reheat'] := 10;
ampl: solve;
presolve: constraint Time['reheat']
  cannot hold: body <= 10 cannot be
  >= 11.25; difference = -1.25
```

エラーの手がかり

- “ignoring integrality” → 整数計画対応ソルバーが必要
- “can't handle nonlinearities” → 非線形対応ソルバーが必要
- 解が遅い／メッセージなしで戻る → ソルバー固有オプションを確認

ソルバーの選択と比較

同じ問題を3つのソルバーで比較

```
ampl: model steelT.mod; data steelT.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
15 iterations, objective 515033

ampl: reset;
ampl: model steelT.mod; data steelT.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 8.0.0: optimal solution;
  objective 515033

ampl: reset;
ampl: model steelT.mod; data steelT.dat;
ampl: option solver snopt;
ampl: solve;
SNOPT 6.1-1: Optimal solution found.
15 iterations, objective 515033
```

使い方の基本

- option solver; で既定ソルバー名を確認
- option solver cplex; のように切り替え
- 多くの問題は solver 設定と solve だけで十分

なぜ毎回 reset するのか

reset なしだと、あるソルバーが見つけた解の情報が次に引き継がれ、比較が不公平になる。同じ初期条件から始めて性能を比較するために reset を挟む。

データの変更：reset data / update data

R reset data

- 指定した集合・パラメータの現在値を削除する
- 以後 solve すると未設定のためエラーになりうる

```ampl
reset data MINREQ, MAXREQ,
   amt, n_min, n_max;
```

U update data

- 新しい値が実際に読まれるまで、古い値は削除されない
- 読み込まれなかった項目は以前の値のまま残る

```ampl
update data MINREQ, MAXREQ,
   amt, n_min, n_max;
```

例：MINREQ・amt・n_min のみ新しい値を読み込んだ場合 — reset data なら MAXREQ・n_max は値を失いエラーになるが、update data なら以前の値のまま維持される。

乱数パラメータの再標本化

avail を正規乱数として定義

```
param avail_mean {STAGE} >= 0;
param avail_variance {STAGE} >= 0;
param avail {s in STAGE} =
   Normal(avail_mean[s], avail_variance[s]);
```

- reset data は乱数で計算されるパラメータを再標本化する
- 標本が変われば解も、最適目的値も変わる
- reset（データではなく全体）を出すと乱数生成器自体がリセットされ、値は最初から繰り返される

steel4r：再標本化のたびに解が変わる

```
ampl: solve;
MINOS 5.5: optimal solution found.
3 iterations, objective 187632.2489
ampl: display avail;
reheat 32.3504
  roll 43.038 ;

ampl: reset data avail;
ampl: solve;
MINOS 5.5: optimal solution found.
4 iterations, objective 158882.901
ampl: display avail;
reheat 32.0306
  roll 32.6855 ;
```

let コマンド

配合モデルの上限を試行錯誤で変更

```
ampl: let f_max["CHK"] := 11;
ampl: solve;
MINOS 5.5: optimal solution found.
1 iterations, objective 73.43818182
```

- keyword let ＋（任意で）添字式 ＋ 代入 という形
- = で定義されていない集合・パラメータの値を直接変更できる
- 変数の現在値も変更でき、別の解を試すのに使える

添字集合をまとめて変更

```
let {j in FOOD} f_max[j] := 8;
let {j in FOOD}
   f_max[j] := 1.1 * f_max[j];
```

! 注意：添字づけへの影響

週数 T のような、他のデータの添字づけに使われる値を let で変えると、既存データの添字が無効になりエラーになりうる（Tdata と T を分けて宣言するのが定石）。

表11-1　丸め関数（抜粋）

`ceil(x)`	x の天井（次に大きい整数）
`floor(x)`	x の床（次に小さい整数）
`round(x, n)`	小数点以下 n 桁に丸め
`trunc(x)`	整数に切り捨て

モデル構成要素の除去と再定義

D

delete

他の構成要素から参照されていなければ取り除く。目的・制約はいつでも削除可

P

purge

指定要素と、それに直接・間接に依存するすべての構成要素をまとめて削除

X

xref

ある構成要素にどの構成要素が依存しているかを調べる

E

redeclare

宣言を改訂版で置き換える。依存要素やデータ値はそのまま保たれる

purge f_min の影響範囲を先に確認

```
ampl: xref f_min;
# 4 entities depend on f_min:
f_max
Buy
Total_Cost
Diet
```

redeclare の例

```
ampl: redeclare param f_min
   {FOOD} > 0 integer;

# f_min の妥当性条件だけを変更
# 依存要素・既存データ値は保持
```

fix・unfix・drop・restore

drop

特定の制約・目的を一時的に無視する

restore

drop の効果を打ち消し、制約を復活させる

fix

変数を現在値に固定する（＝制約であるかのように扱う）

unfix

fix の効果を打ち消す

いずれもキーワード＋任意の添字式＋名前という共通構文。*drop/restore* は制約・目的、*fix/unfix* は変数を対象とする。*連続する drop の効果は累積する。*

fix の実例（配合モデル）

```
ampl: let {j in FOOD} Buy[j] := f_min[j];
ampl: fix {j in FOOD: amt["NA",j]>1200}
        Buy[j];
ampl: solve;
MINOS 5.5: optimal solution found.
7 iterations, objective 86.92

# ナトリウム過多の食品を固定し
# 残りの変数だけを最適化
```

添字式・代入句も併用可能

```
drop {i in MAXNOT} Diet_Max[i];
fix {j in FOOD:...} Buy[j] := f_min[j];
```

整数性の緩和

0/1 relax_integrality オプション

```ampl
option relax_integrality 1;
```

- 既定 0 → 1 にすると整数制限をすべて無視
- integer 変数は元の上下限のまま連続値に
- binary 変数は下限0・上限1の連続値に

.r .relax 接尾辞

```
let {j in FOOD: f_min[j]>allow_frac}
   Buy[j].relax := 1;
```

AMPL の緩和 vs ソルバーの緩和指示

scale=0.95、f_max=10 の配合モデル（整数変数）で比較：

緩和なし

objective 122.89

上限9.5を整数条件で9に前処理で切り下げ

ソルバーの relax 指示

objective 120.24

前処理は9のまま。ソルバー側だけ緩和

relax_integrality 1

objective 119.15

前処理も9.5のまま。最も緩い解

SUMMARY

第11章のまとめ

1	**対話環境の基本**	ampl: プロンプト、セミコロン終端、option コマンドによる振る舞いの制御
2	**model・data・solve**	現在のモデルを積み上げ、objective で目的を選び、solver でソルバーを切り替える
3	**データの再設定**	reset data／update data で値を入れ替え、let で個別値をピンポイントに変更
4	**モデルの限定変更**	delete・purge・redeclare で構成要素を、fix・drop で一時的な制約緩和を行う
5	**整数性の制御**	relax_integrality と .relax 接尾辞で、必要な変数だけ整数条件を緩められる

次章：第12章へ続く