AMPL：数理計画のためのモデリング言語

第12章

表示コマンド

display・*print*・*printf* で結果を読み解き、思いどおりに整形する

集合・値の表示 | 書式と数値オプション | 解の双対値・被約費用 | モデル検査 (show/xref/expand)

Fourer, Gay, Kernighan 著（日本語版）

AGENDA

本章の内容

12章

12.1 結果を眺める：display コマンド

12.2 display の書式オプション

12.3 display の数値オプション

12.4 その他の出力コマンド：print と printf

12.5 解に関連する値

12.6 モデルと問題例に関するその他の表示機能

12.7 出力を操作する一般的な機能

結果を眺める：display コマンド

基本

- display に見たいものの記述を続けて打ち込むだけで、値を見慣れた配置に自動整形する。
- 第9章のデータ文と同じリスト・表の形式を可能なかぎり使う。
- 集合の内容は display と集合名のリストで表示できる。
- 添字づけられた集まりの各集合も一括表示できる（例：PROD, AREA）。
- 引数は宣言済みの集合でなくてよく、集合に評価される任意の式を渡せる。

対話例（*amplsession*）

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

ampl: display union {p in PROD} AREA[p];
set union {p in PROD} AREA[p] :=
   east north west export;
```

パラメータと変数の表示

次元別

1 1次元データ

単純な集合で添字づけられたパラメータ・変数は列（リスト）形式。例：avail

2 2次元データ

対の集合・二つの集合で添字づけられた値。少なければリスト、多ければ表になる。

3+ 3次元以上

AMPLは二次元の表に「スライス」して提示する。表の数が最小になる軸を自動選択。

Σ スカラー・式

display T や display sin(1)^2+cos(1)^2 のように単一値もそのまま表示できる。

対話例：1次元→2次元→3次元(スライス)

```
ampl: display avail;
avail [*] :=
reheat 35
  roll 40 ;

ampl: display supply;
supply :=
CLEV bands  700
CLEV coils 1600
GARY bands  400  ... ;

ampl: display Trans;
Trans [CLEV,*,*]
: bands coils plate :=
DET   0   750    0  ...
```

添字づけられた式の表示

式の表示

- 任意の算術式を display できるが、配列同士の演算（revenue * Sell）は認識されない。
- 式の添字づけられた集まりを表示するには、添字づけを明示的に指定する。
- 添字式＋(式のリスト) は1つの「表示項目」として扱われる。
- display コマンド全体を添字づければ、添字集合の各要素ごとに個別出力できる（表のスライスの並べ替えに有用）。

対話例（誤り例→正しい書き方）

```
ampl: display revenue * Sell;
syntax error

ampl: display {p in PROD, t in 1..T}
ampl?    revenue[p,t]*Sell[p,t];
revenue[p,t]*Sell[p,t] [*,*] (tr)
:   bands    coils :=
1   150000    9210
2   156000   87500 ...

ampl: display {p in PROD}:
ampl?   {i in ORIG,j in DEST} Trans[i,j,p];
Trans[i,j,'bands'] [*,*] (tr) ...
```

display の書式オプション（表12-1）

表12-1

オプション	意味（既定値）
`display_1col`	表をリスト形式で表示する要素数の上限（20）
`display_transpose`	行数 − 列数 ＜ display_transpose なら表を転置（0）
`display_width`	1行あたりの最大文字数（79）
`gutter_width`	表の列間の空白幅（3）
`omit_zero_cols`	非0なら、全て零の列を表示から省く（0）
`omit_zero_rows`	非0なら、全て零の行を表示から省く（0）

リストと表の配置・転置

配置

display_1col：0でコンパクト表示を強制

```
ampl: option display_1col 0;
ampl: display required;
required [*] :=
Fri1 100  Mon1 100  Sat1 100 ...
Fri2  78  Mon2  78  Sat2  78 ...
Fri3  52  Mon3  52  Thu1 100 ...;
```

display_transpose：*正*=転置優勢／負=抑制

```
ampl: option display_transpose -5;
ampl: display cost;
cost [*,*]
:     DET FRA FRE LAF LAN STL WIN :=
CLEV    9  27   .   17  12  26   9
GARY   14   .   .    8  11  16   .
PITT    .  24  99    .   .  28  13 ;
```

- display_1col：表示値の個数がこの値以下なら1列リスト、超えればコンパクトな形式（表）を使う。既定20。
- 表の向きは既定で「行 ＞ 列」を優先。列数が行数を超える場合は転置され、先頭行に (tr) が付く。
- display_transpose：規則は「行数 − 列数 ＜ display_transpose のときのみ転置」。

0 行幅の制御と零の抑制

幅と零

行幅の制御

- display_width：1行の最大文字数（既定79）。超える表は縦に2つ以上の表へ分割される。
- 列見出しが値よりずっと広い場合は $1, $2 のような略号が自動導入される。
- 見出しが値より狭ければ gutter_width（列間の空白、既定3）を減らして詰められる。

零の抑制

- omit_zero_rows を1にすると、全て零の行が表示から省かれる。
- omit_zero_cols を1にすると、全て零の列が表示から省かれる。
- 割当問題のようにほぼ0の変数が多い場合に、関心のある項目だけに絞り込める。

対話例：零を含む大きな表を圧縮する

```
ampl: option omit_zero_rows 1;
ampl: display Trans;
Trans :=
Coullard C118  1
Daskin   D241  1
Hazen    C246  1  ... ;
```

display の数値オプション（表12-2）

表12-2

オプション	意味（既定値）
`display_eps`	零と区別して表示する最小絶対値（0）
`display_precision`	表示数値を丸める有効桁数。0なら完全精度（6）
`display_round`	小数点位置での丸め桁。display_precision に優先（空）
`solution_precision`	解の値そのものを丸める有効桁数（0）
`solution_round`	解の値そのものを丸める桁位置。solution_precision に優先（空）

見え方だけを変える： display_eps / display_precision / display_round　**実際の値を変える：** solution_precision / solution_round（solve の前に設定）

数値の見え方と丸め

丸め

precision（有効桁数）と round（丸め位置）

```
ampl: option display_precision 3;
Make['bands',4]/rate['bands'] = 32.1

ampl: option display_precision 0;
...  = 32.14285714285713

ampl: option display_round 2;
:  bands  coils :=
1  29.95  10.05
4  32.14   7.86 ;
```

- display_precision：既定6桁。0なら計算機表現から必要十分な桁数で正確に表す。
- display_round：小数点以下の桁数を固定（非負）／百・十の位に丸め（負）。整数値なら precision に優先。
- display_eps：この絶対値未満は厳密な零として扱い、1e-17 のような微小残差の表示を防ぐ。
- **注意：これらは見え方のみを変え、solve 後の実際の解の値には影響しない。**

その他の出力コマンド：print と printf

整形出力

print：整形しない1行出力

```
ampl: print {p in PROD} (p, rate[p]);
bands 200 coils 140

ampl: print {t in 1..T}: {p in PROD} Make[p,t];
5990 1407
6000 1400 ...
```

- 集合は引数にできないが、その要素は渡せる。添字づけを入れ子にできる点が display と異なる。
- print_separator でスペース以外の区切り（タブ等）を指定でき、表計算への出力に便利。

printf：C言語風の書式指定

```
ampl: printf "Total revenue is $%6.2f.\n",
ampl?    sum {p in PROD,t in 1..T}
ampl?       revenue[p,t]*Sell[p,t];
Total revenue is $787810.00.
```

- 最初の印字項目が書式文字列。%6.2f のような変換指定で幅・小数点以下の桁数を制御する。
- 改行は \n で明示。添字式＋コロンを使えば要素ごとに1行を自動生成できる。
- 変換指定の数と値の数が一致する必要があり、列数が変わる表の出力には不向き。

目的関数・上下限・スラック

接尾辞

- AMPLは「変数名／制約名 ＋ ドット ＋ 接尾辞」の修飾名で、最適解に付随する量を参照する。
- 目的関数の名前はそのまま最適値を指す（display、print、printf で利用可）。
- .lb / .ub 上下限（AMPLの前処理が導いた上下限を含む）
- .slack 値と近いほうの限界との差
- .body 制約本体（下限 ≦ 本体 ≦ 上限）の現在値

対話例：上下限・本体・スラックの表示

```
ampl: display Buy.lb, Buy, Buy.ub, Buy.slack;
:    Buy.lb   Buy   Buy.ub Buy.slack :=
BEEF    2      2       10       0
CHK     2     10       10       0
MTL     2   6.236      10   3.764
SPG     2   5.258      10   3.258 ;

ampl: display Diet_Min.lb, Diet_Min.body,
ampl?   Diet_Min.ub;
:   .lb    .body     .ub :=
A   700  1013.98  Infinity
C   700   700.00  Infinity ;
```

双対値と被約費用

感度

- 制約の双対値（潜在価格）：修飾接尾辞なしの制約名で参照。制約の右辺（限界）が1単位変化したときの目的値の変化率を表す。
- スラックが正の制約では双対値は零。スラックが零のとき、双対値は意味をもつ。
- 変数側の対応物は被約費用（.rc）。上下限のいずれかで値が止まっている変数で非零になりうる。
- 目的値を制約の定数項に対して図示すると、区分的線形の曲線になる（右図）。双対値・被約費用はその曲線の傾き＝局所的な感度を1つだけ与える。

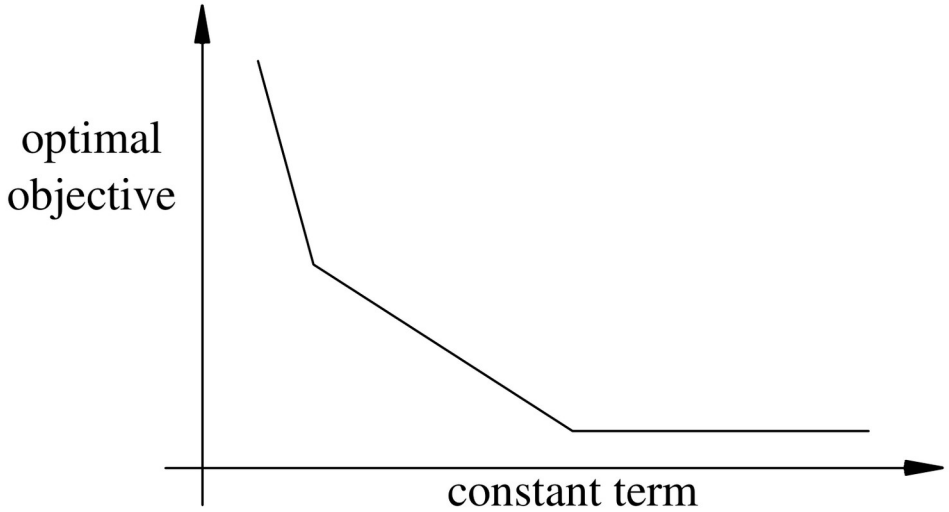

図12-1：目的関数の区分的線形なグラフ
縦軸＝最適な目的値、横軸＝制約の定数項（限界）

モデル構成要素と問題例の検査

検査コマンド

show — モデル構成要素の一覧・宣言の表示

単独で全構成要素名を種類別に一覧。show Total_Cost のように個別の宣言も表示でき、モデルファイルを探す手間を省く。

xref — 依存関係の表示

指定した構成要素に直接・間接に依存するすべての構成要素（制約・目的など）を一覧する。

expand — 生成された問題例の表示

モデルとデータから実際に生成された制約・目的を展開表示。solexpand は前処理後の「ソルバーの視点」を示す。

対話例：show / xref / expand

```
ampl: show;
parameters: demand fcost limit ...
variables:  Trans  Use
constraints: Demand Max_Serve ...

ampl: xref demand, Trans;
# 2 entities depend on demand:
check 1
Demand

ampl: expand Supply;
subject to Supply['GARY']:
  Trans['GARY','FRA'] + ... = 1400;
```

総称的な別名と資源の一覧

総称名

変数・制約・目的の総称的な別名

- _nvars / _ncons / _nobjs：現在の問題の変数・制約・目的の数。
- _varname[1.._nvars] など：各構成要素の真のAMPL名を返す文字列パラメータ。
- _var[j] / _con[j] / _obj[j]：番号で構成要素を参照する別名。_var[5].rc のように接尾辞も使える。
- 接頭辞を _s に置き換えると（_svar 等）、前処理後の「ソルバーの視点」を参照できる。

資源の一覧

- show_stats：solve 時に問題の規模（変数・制約数、非零要素数など）を要約表示。
- times：AMPL翻訳系の所要時間とメモリの要約を表示。
- gentimes：genmod 段階の詳細な資源使用量。処理が遅い原因の特定に有用。
- _solve_time / _ampl_time：直近の solve に要したCPU秒数を保持する定義済みパラメータ。

対話例：上限に達していない変数だけを列挙

```
ampl: display {j in 1.._nvars:
ampl?  _var[j] < _var[j].ub - 0.00001} _varname[j];
_varname[j] [*] :=
2 "PD_Ship['SE']"          5 "DW_Ship['NE','BWI']"
6 "DW_Ship['SE','EWR']"    7 "DW_Ship['SE','BWI']" ;
```

出力を操作する一般的な機能

出力制御

リダイレクト（>／>>）

display / print / expand などのコマンド末尾に >filename を付ければ結果をファイルに保存。>> は追記、close で明示的に閉じる。

log_file オプション

以降のコマンドと応答をまとめて記録。log_model / log_data で入力ファイルの内容も記録に含められる。

メッセージの制限

eexit（既定-10）でエラー時の打ち切り方を制御。presolve_warnings（既定5）、bad_subscripts（既定3）で警告数を制限。

対話例：リダイレクト／記録／打ち切り

```
ampl: display supply >multi.out;
ampl: close multi.out;

ampl: option log_file 'multi.tmp';

ampl: option eexit -3;
ampl: model diet.mod;
ampl: data diet.mod;
diet.mod, line 4 (offset 32):
  expected ; ( [ : or symbol
Bailing out after 3 warnings.
```

SUMMARY

✓ 本章のまとめ

1	**display**	集合・パラメータ・変数・任意の式を、1次元はリスト、2次元以上は表やスライスへ自動整形して表示する。
2	**書式・数値オプション**	display_1col / display_transpose / display_width / omit_zero_* で配置を、display_precision / display_round / display_eps で見え方を制御する。solution_* は実際の値を丸める。
3	**print と printf**	整形しない出力と、C言語風の書式で完全に制御した出力を使い分け、他ツール向けデータや報告書を作る。
4	**解に付随する値**	.lb / .ub / .slack / .body、制約の双対値、変数の被約費用（.rc）で最適解を深く解釈できる。
5	**モデル検査と出力制御**	show / xref / expand でモデルと問題例を検査し、> によるリダイレクトや log_file、eexit 等で出力とメッセージ量を管理する。

次章：第13章へ続く