Numerical Issues and Influences in the Design of Algebraic Modeling Languages for Optimization

Robert Fourer
Department of Industrial Engineering & Management Sciences
Northwestern University

David M. Gay
AMPL Optimization LLC

20th Biennial Conference on Numerical Analysis
University of Dundee, Scotland — 24-27 June 2003

Abstract

The idea of a modeling language is to describe mathematical problems in a symbolic form that is familiar to people, but that can be processed by computer systems. In particular the concept of an algebraic modeling language, based on objective and constraint expressions in terms of decision variables, has proved to be valuable for a broad range of optimization and related problems.

One modeling language can work with numerous solvers, each of which implements one or more optimization algorithms. The separation of model specification from solver execution is thus a key tenet of modeling language design. Nevertheless, several issues in numerical analysis that are critical to solvers are also important in implementations of modeling languages. So-called presolve procedures, which tighten bounds with the aim of eliminating some variables and constraints, are numerical algorithms that require carefully chosen tolerances and can benefit from directed roundings. Correctly rounded binary-decimal conversion is valuable in portably conveying problem instances and in debugging. Further rounding options offer tradeoffs between accuracy, convenience, and readability in displaying numerical data.

Modeling languages can also strongly influence the development of solvers. Most notably, for smooth nonlinear optimization, the ability to provide numerically computed, exact first and second derivatives has made modeling languages a valuable tool in solver development. The generality of modeling languages has also encouraged the development of more general solvers, such as for optimization problems with equilibrium constraints.

This presentation draws from our experience in developing the AMPL modeling language to provide examples in all of the above areas. We conclude by describing possibilities for future work that would have a significant numerical aspect.
Outline

**Rounding and conversion**
- Displayed vs. actual values
- Correctly rounded conversions

**Presolving**
- Fixed variables, redundant constraints
- Infeasible constraints

**Influence on solvers**
- Second derivatives ↔ IP solvers
- Complementarity problems ↔ MPECs

**Future influences**
- Quadratic expressions
- Matrix functions and constraints
- Nonlinear expressions as input to solvers

A Brief Introduction to AMPL:  
The McDonald’s Diet Problem

<table>
<thead>
<tr>
<th>Foods</th>
<th>Nutrients</th>
</tr>
</thead>
<tbody>
<tr>
<td>QP Quarter Pounder</td>
<td>Prot Protein</td>
</tr>
<tr>
<td>FR Fries, small</td>
<td>Iron Iron</td>
</tr>
<tr>
<td>MD McLean Deluxe</td>
<td>VitA Vitamin A</td>
</tr>
<tr>
<td>SM Sausage McMuffin</td>
<td>Calcs Calories</td>
</tr>
<tr>
<td>BM Big Mac</td>
<td>VitC Vitamin C</td>
</tr>
<tr>
<td>1M 1% Lowfat Milk</td>
<td>Carb Carbohydrates</td>
</tr>
<tr>
<td>FF Filet-O-Fish</td>
<td>Calc Calcium</td>
</tr>
<tr>
<td>OJ Orange Juice</td>
<td></td>
</tr>
<tr>
<td>MC McGrilled Chicken</td>
<td></td>
</tr>
</tbody>
</table>
McDonald’s Diet Problem Data

<table>
<thead>
<tr>
<th></th>
<th>QP</th>
<th>MD</th>
<th>BM</th>
<th>FF</th>
<th>MC</th>
<th>FR</th>
<th>SM</th>
<th>1M</th>
<th>OJ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost</strong></td>
<td>1.8</td>
<td>2.2</td>
<td>1.8</td>
<td>1.4</td>
<td>2.3</td>
<td>0.8</td>
<td>1.3</td>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td><strong>Protein</strong></td>
<td>28</td>
<td>24</td>
<td>25</td>
<td>14</td>
<td>31</td>
<td>3</td>
<td>15</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td><strong>Vitamin A</strong></td>
<td>15</td>
<td>15</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td><strong>Vitamin C</strong></td>
<td>6</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td><strong>Calcium</strong></td>
<td>30</td>
<td>20</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>20</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td><strong>Iron</strong></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>15</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td><strong>Calories</strong></td>
<td>510</td>
<td>370</td>
<td>500</td>
<td>370</td>
<td>400</td>
<td>220</td>
<td>345</td>
<td>110</td>
<td>80</td>
</tr>
<tr>
<td><strong>Carbo</strong></td>
<td>34</td>
<td>35</td>
<td>42</td>
<td>38</td>
<td>42</td>
<td>26</td>
<td>27</td>
<td>12</td>
<td>20</td>
</tr>
</tbody>
</table>

Formulation: Too General

Minimize \( cx \)

Subject to \( Ax = b \)

\( x \geq 0 \)
Formulation: Too Specific

\[ \text{Minimize} \quad 1.84 x_{QP} + 2.19 x_{MD} + 1.84 x_{BM} + 1.44 x_{FF} + 2.29 x_{MC} + 0.77 x_{FR} + 1.29 x_{SM} + 0.60 x_{1M} + 0.72 x_{OJ} \]

Subject to

\[ 28 x_{QP} + 24 x_{MD} + 25 x_{BM} + 14 x_{FF} + 31 x_{MC} + 3 x_{FR} + 15 x_{SM} + 9 x_{1M} + 1 x_{OJ} \geq 55 \]
\[ 15 x_{QP} + 15 x_{MD} + 6 x_{BM} + 2 x_{FF} + 8 x_{MC} + 2 x_{FR} + 9 x_{SM} + 2 x_{1M} + 2 x_{OJ} \geq 100 \]
\[ 6 x_{QP} + 10 x_{MD} + 15 x_{BM} + 15 x_{FF} + 6 x_{MC} + 8 x_{FR} + 15 x_{SM} + 10 x_{1M} + 4 x_{OJ} \geq 100 \]
\[ 30 x_{QP} + 20 x_{MD} + 15 x_{BM} + 0 x_{FF} + 15 x_{MC} + 8 x_{FR} + 15 x_{SM} + 20 x_{1M} + 30 x_{OJ} \geq 100 \]
\[ 20 x_{QP} + 20 x_{MD} + 20 x_{BM} + 10 x_{FF} + 8 x_{MC} + 2 x_{FR} + 15 x_{SM} + x_{1M} + 2 x_{OJ} \geq 100 \]
\[ 510 x_{QP} + 370 x_{MD} + 500 x_{BM} + 370 x_{FF} + 400 x_{MC} + 220 x_{FR} + 345 x_{SM} + 110 x_{1M} + 80 x_{OJ} \geq 2000 \]
\[ 34 x_{QP} + 35 x_{MD} + 42 x_{BM} + 38 x_{FF} + 42 x_{MC} + 26 x_{FR} + 27 x_{SM} + 12 x_{1M} + 20 x_{OJ} \geq 350 \]

Formulation: Algebraic Model

Given \( F \), a set of foods

\( N \), a set of nutrients

and

\( a_{ij} \geq 0 \), the units of nutrient \( i \) in one serving of food \( j \),

for each \( i \in N \) and \( j \in F \)

\( b_i > 0 \), the units of nutrient \( i \) required, for each \( i \in N \)

\( c_j > 0 \), the cost per serving of food \( j \), for each \( j \in F \)

Define \( x_j \geq 0 \), the number of servings of food \( j \) to be purchased, for each \( j \in F \)

Minimize \( \sum_{j \in F} c_j x_j \)

Subject to \( \sum_{j \in F} a_{ij} x_j \geq b_i \), for each \( i \in N \)
Algebraic Model in AMPL

```AMPL
set NUTR;   # nutrients
set FOOD;   # foods

param amt (NUTR, FOOD) >= 0;   # amount of nutrient in each food
param nutrLow (NUTR) >= 0;    # lower bound on nutrients in diet
param cost (FOOD) >= 0;       # cost of foods

var Buy (FOOD) >= 0 integer;  # amounts of foods to be purchased

minimize TotalCost: sum {j in FOOD} cost[j] * Buy[j];

subject to Need {i in NUTR}:
    sum {j in FOOD} amt[i, j] * Buy[j] >= nutrLow[i];
```

Data for the AMPL Model

```AMPL
param: FOOD: cost :=
  "Quarter Pounder"  1.84 "Fries, small"  .77
  "McLean Deluxe"  2.19 "Sausage McMuffin" 1.29
  "Big Mac"  1.84 "1% Lowfat Milk"  .60
  "Filet-O-Fish"  1.44 "Orange Juice"  .72
  "McGrilled Chicken"  2.29 ;

param: NUTR: nutrLow :=
  Prot  55   VitA 100   VitC  100
  Calc 100   Iron 100   Cals 2000   Carb 350 ;

param amt (tr):
  "Quarter Pounder"  510 34 28 15 6 30 20
  "McLean Deluxe"  370 35 24 15 10 20 20
  "Big Mac"  500 42 25 6 2 25 20
  "Filet-O-Fish"  370 38 14 2 0 15 10
  "McGrilled Chicken"  400 42 31 8 15 15 8
  "Fries, small"  220 26 3 0 15 0 2
  "1% Lowfat Milk"  110 12 9 10 4 30 0
  "Orange Juice"  80 20 1 2 120 2 2 ;
```
Continuous-Variable Solution

```
ampl: model mcdiet1.mod;
ampl: data mcdiet1.dat;
ampl: solve;
MINOS 5.5: ignoring integrality of 9 variables
MINOS 5.5: optimal solution found.
7 iterations, objective 14.8557377
ampl: display Buy;
Buy [*] :=
  1% Lowfat Milk 3.42213
  Big Mac 0
  Filet-O-Fish 0
  Fries, small 6.14754
  McGrilled Chicken 0
  McLean Deluxe 0
  Orange Juice 0
  Quarter Pounder 4.38525
  Sausage McMuffin 0
```

Integer-Variable Solution

```
ampl: option solver cplex;
ampl: solve;
CPLEX 8.1.0: optimal integer solution; objective 15.05
27 MIP simplex iterations
15 branch-and-bound nodes
ampl: display Buy;
Buy [*] :=
  1% Lowfat Milk 4
  Big Mac 0
  Filet-O-Fish 1
  Fries, small 5
  McGrilled Chicken 0
  McLean Deluxe 0
  Orange Juice 0
  Quarter Pounder 4
  Sausage McMuffin 0
```
Same for 63 Foods, 12 Nutrients

```
ampl: reset data;
ampl: data mcdiet2.dat;
ampl: option solver minos;
ampl: solve;
MINOS 5.5: ignoring integrality of 63 variables
MINOS 5.5: optimal solution found.
16 iterations, objective -1.786806582e-14
ampl: option omit_zero_rows 1;
ampl: display Buy;
Buy [*] :=
    Bacon Bits 55
    Barbeque Sauce 50
    Hot Mustard Sauce 50
```

Revised Algebraic Model in AMPL

```
set NUTR ordered;
set FOOD ordered;
param cost {FOOD} >= 0;
param f_min {FOOD} >= 0, default 0;
param f_max {j in FOOD} >= f_min[j], default Infinity;
param n_min {NUTR} >= 0, default 0;
param n_max {i in NUTR} >= n_min[i], default Infinity;
param amt {NUTR,FOOD} >= 0
var Buy {j in FOOD} integer >= f_min[j], <= f_max[j];
minimize Total_Cost: sum {j in FOOD} cost[j] * Buy[j];
minimize Nutr_Amt {i in NUTR}: sum {j in FOOD} amt[i,j] * Buy[j];
subject to Diet {i in NUTR}:
    n_min[i] <= sum {j in FOOD} amt[i,j] * Buy[j] <= n_max[i];
```
Revised Algebraic Model in AMPL (cont’d)

subject to McNuggetsSauces:
    Buy["Hot Mustard Sauce"] + Buy["Barbeque Sauce"] +
    Buy["Sweet 'N Sour Sauce"] + Buy["Honey"]
<= 2 * Buy["Chicken McNuggets (6 pcs)"] +
    3 * Buy["Chicken McNuggets (9 pcs)"] +
    6 * Buy["Chicken McNuggets (20 pcs)"];

subject to SaladToppings:
    Buy["Croutons"] + Buy["Bacon Bits"]
<= Buy["Chef Salad"] + Buy["Chunky Chicken Salad"] +
    Buy["Garden Salad"] + Buy["Side Salad"];  

subject to SaladDressings:
    Buy["Bleu Cheese Dressing"] + Buy["Ranch Dressing"] +
    Buy["1000 Island Dressing"] + Buy["Lite Vinaigrette Dressing"] +
    Buy["French Rdcd Cal Dressing"]
<= Buy["Chef Salad"] + Buy["Chunky Chicken Salad"] +
    Buy["Garden Salad"] + Buy["Side Salad"];  

subject to OneDrinkPerMeal:
    3 = Buy["Vanilla Shake"] + Buy["Chocolate Shake"] +
        Buy["Strawberry Shake"] +
        Buy["1% Lowfat Milk"] + Buy["Orange Juice"] +
        Buy["Coca-Cola (small)"] + Buy["Coca-Cola (medium)"] +
        Buy["Coca-Cola (large)"] +
        Buy["Diet Coke (small)"] + Buy["Diet Coke (medium)"] +
        Buy["Diet Coke (large)"] +
        Buy["Sprite (small)"] + Buy["Sprite (medium)"] +
        Buy["Sprite (large)"] +
        Buy["H-C Orange Drink (small)"] +
        Buy["H-C Orange Drink (medium)"] +
        Buy["H-C Orange Drink (large)"];  

subject to FatCaloriesLimit:
    sum {j in FOOD} amt["CalFat",j] * Buy[j]
<= 0.3 * sum {j in FOOD} amt["Cal",j] * Buy[j];
Revised Solution (at most 2 of every food)

```ampl
reset;
model mcdiet2.mod;
data mcdiet2all.dat;
option solver cplex;
solve;
CPLEX 8.1.0: optimal integer solution; objective 8.86
511 MIP simplex iterations
325 branch-and-bound nodes
display Buy;
Buy [*] :=
    Cheerios 1
    Cheeseburger 2
    'H-C Orange Drink (large)' 1
    Hamburger 2
    'Orange Juice' 1
    'Raspberry Danish' 1
    'Side Salad' 1
    'Strawberry Shake' 1
```

Essential Modeling Language Features

- **Sets and indexing**
  - Simple sets
  - Compound sets
  - Computed sets

- **Variables, objectives and constraints**
  - Linear, piecewise-linear
  - Nonlinear
  - Integer

- and much more...
  - Express problems in the various ways that people do
  - Support a broad variety of modeling situations
  - Drive varied solvers
Modeling Language Features (cont’d)

Programming iterative schemes
   Loops over sets, if-then-else tests
   Switching between subproblems
   Debugging

Representing other types of models
   Complementarity problems
   General combinatorial problems (to come)
   Stochastic programs (to come)

Communicating with other systems
   Relational database access
   Internet optimization services
   Solver-specific directives, results & diagnostic information

Commercial Modeling Languages

AIMMS  www.aimms.com
AMPL   www.ampl.com
GAMS   www.gams.com
LINGO  www.lindo.com
MPL    www.maximal-usa.com
OPL    www.ilog.com/products/oplstudio/
Airline Fleet Assignment

set FLEETS;
param fleet_size (FLEETS) >= 0;
set CITIES;
set TIMES circular;
set FLEET_LEGS within 
   {f in FLEETS, c1 in CITIES, t1 in TIMES, 
   c2 in CITIES, t2 in TIMES: c1 <> c2 and t1 <> t2};
   # (f,c1,t1,c2,t2) represents the availability of fleet f 
   # to cover the leg that leaves c1 at t1 and 
   # whose arrival time plus turnaround time at c2 is t2
param leg_cost (FLEET_LEGS) >= 0;

Computed Sets

set LEGS := setof {{f,c1,t1,c2,t2} in FLEET_LEGS} {c1,t1,c2,t2};
   # the set of all legs that can be covered by some fleet
set SERV_CITIES {f in FLEETS} :=
   union {{f,c1,c2,t1,t2} in FLEET_LEGS} {c1,c2};
   # for each fleet, the set of cities that it serves
set OP_TIMES {f in FLEETS, c in SERV_CITIES{f}} circular by TIMES :=
   setof {{f,c,c2,t1,t2} in FLEET_LEGS} t1 union
   setof {{f,c1,c,t1,t2} in FLEET_LEGS} t2;
   # for each fleet and city served by that fleet,
   # the set of active arrival & departure times at that city,
   # with arrival time padded for turn requirements
Underlying Network Model

minimize Total_Cost;

node Balance \{f \in FLEETS, c \in SERV_CITIES[f], OP_TIMES[f,c]\};
# for each fleet and city served by that fleet,
# a node for each possible time

arc Fly \{(f,c1,t1,c2,t2) \in FLEET_LEGS\} \geq 0, \leq 1,
from Balance[f,c1,t1], to Balance[f,c2,t2],
obj Total_Cost leg_cost[f,c1,t1,c2,t2];
# arcs for fleet/flight assignments

arc Sit \{f \in FLEETS, c \in SERV_CITIES[f], t \in OP_TIMES[f,c]\} \geq 0,
from Balance[f,c,t], to Balance[f,c,next(t)];
# arcs for planes on the ground

subj to Service \{(c1,t1,c2,t2) \in LEGS\}:
sum \{(f,c1,t1,c2,t2) \in FLEET_LEGS\} Fly[f,c1,t1,c2,t2] = 1;
# each leg must be served by some fleet

subj to FleetSize \{f \in FLEETS\}:
sum \{(f,c1,t1,c2,t2) \in FLEET_LEGS:\
ord(t2, TIMES) < ord(t1, TIMES)\} Fly[f,c1,t1,c2,t2] +
sum \{c \in SERV_CITIES[f]\} Sit[f,c,last(OP_TIMES[f,c])] \leq fleet_size[f];
# number of planes used is the number in the air at the
# last time (arriving "earlier" than they leave)
# plus the number on the ground at the last time in each city
Rounding and Conversion

Rounding
Display of zeros
Number of displayed digits
Number of actual digits

Conversion
Correctly rounded binary $\leftrightarrow$ decimal conversion
"Maximum" precision

Display “epsilon” production-transportation model

```plaintext
var Make {ORIG,PROD} >= 0;       # tons produced at origins
var Trans {ORIG,DEST,PROD} >= 0; # tons shipped

minimize Total_Cost:
    sum {i in ORIG, p in PROD} make_cost[i,p] * Make[i,p] +
    sum {i in ORIG, j in DEST, p in PROD} trans_cost[i,j,p] * Trans[i,j,p];

subject to Time {i in ORIG}:
    sum {p in PROD} (1/rate[i,p]) * Make[i,p] <= avail[i];

subject to Supply {i in ORIG, p in PROD}:
    sum {j in DEST} Trans[i,j,p] = Make[i,p];

subject to Demand {j in DEST, p in PROD}:
    sum {i in ORIG} Trans[i,j,p] = demand[j,p];
```
Display "epsilon" (cont'd)

```
ampl: model steelP.mod;
ampl: data steelP.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
27 iterations, objective 1392175
ampl: display Make;
Make [*,*] :  bands   coils   plate :=
CLEV 1.91561e-14 1950 3.40429e-14
GARY 1125 1750 300
PITT 775 500 500
;
ampl: option display_eps 1e-10;
ampl: display Make;
Make [*,*] :  bands   coils   plate :=
CLEV 0 1950 0
GARY 1125 1750 300
PITT 775 500 500
;
```

Display Precision
economic equilibrium model with price-sensitive demands

```
set PROD;   # products
set ACT;    # activities
param cost {ACT} > 0;      # cost per unit of each activity
param io {PROD,ACT} >= 0;  # units of each product from # 1 unit of each activity
param demzero {PROD} > 0;  # intercept and slope of the demand
param demrate {PROD} >= 0; # as a function of price
var Price {i in PROD};
var Level {j in ACT};

subject to Pri_Compl {i in PROD}:
    Price[i] >= 0 complements
    sum {j in ACT} io[i,j] * Level[j]  >= demzero[i] - demrate[i] * Price[i];

subject to Lev_Compl {j in ACT}:
    Level[j] >= 0 complements
    sum {i in PROD} Price[i] * io[i,j] <= cost[j];
```
Display Precision (cont’d)

```ampl
ampl: model econnl.mod;
ampl: data econnl.dat;
ampl: solve;
Job has been submitted to Kestrel
Kestrel/NEOS Job number : 273481
Kestrel/NEOS Job password : ymDgiDn
Check the following URL for progress report :
......
Path v4.5: Solution found.
13 iterations (5 for crash); 10 pivots.
20 function, 14 gradient evaluations.
ampl: option omit_zero_rows 1;
ampl: display Price;
Price [*] :=
 AA1  16.7051
 AC1   5.44585
 BC1  48.909
 BC2  8.90899
;
```

Display Precision (cont’d)

```ampl
ampl: option display_precision 4;
ampl: display Level;
Level [*] :=
P1a  450.7
P3  190.1
P3c  1789
;
ampl: option display_precision 9;
ampl: display Level;
Level [*] :=
P1a  450.681489
P3  190.123755
P3c  1789.33403
;
ampl: option display_precision 0;  # "maximum" precision
ampl: display Level;
Level [*] :=
P1a  450.68148928230426
P3  190.1237550901998
P3c  1789.3340267897368
;
```
Display Rounding

ampl: display Price;
Price [*] :=
AA1 16.7051
AC1 5.44585
BC1 48.909
BC2 8.90899
;
ampl: option display_round 2;
ampl: display Price;
Price [*] :=
AA1 16.71
AC1 5.45
BC1 48.91
BC2 8.91
;
ampl: option display_round 0;
ampl: display Price;
Price [*] :=
AA1 17
AC1 5
BC1 49
BC2 9
;

Maximum Precision

Correctly rounded decimal-to-binary conversion

Binary value “closest” to a given decimal number, for given binary representation and rounding sense
Clinger (1990) uses IEEE double-extended arithmetic
Gay (1990) adapts to use double-precision arithmetic

Correctly rounded binary-to-decimal conversion

Shortest decimal number that yields a given binary number when correctly rounded back to the given precision
Variants for given number of digits or digits after decimal point
Proposed by Steele and White (1990), speeded by Gay (1990)
Maximum Precision: Efficiency

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.23</td>
<td>31</td>
<td>29</td>
<td>109</td>
<td>21</td>
<td>77</td>
</tr>
<tr>
<td>1.23e+20</td>
<td>33</td>
<td>31</td>
<td>137</td>
<td>25</td>
<td>71</td>
</tr>
<tr>
<td>1.23e–20</td>
<td>33</td>
<td>31</td>
<td>195</td>
<td>26</td>
<td>77</td>
</tr>
<tr>
<td>1.23456789</td>
<td>30</td>
<td>37</td>
<td>261</td>
<td>17</td>
<td>78</td>
</tr>
<tr>
<td>1.2345689e+20</td>
<td>32</td>
<td>38</td>
<td>285</td>
<td>22</td>
<td>85</td>
</tr>
<tr>
<td>1.23456789e–20</td>
<td>32</td>
<td>39</td>
<td>415</td>
<td>22</td>
<td>76</td>
</tr>
<tr>
<td>1234565</td>
<td>47</td>
<td>53</td>
<td>36</td>
<td>26</td>
<td>68</td>
</tr>
<tr>
<td>1.234565</td>
<td>131</td>
<td>268</td>
<td>210</td>
<td>17</td>
<td>76</td>
</tr>
<tr>
<td>1.234565e+20</td>
<td>157</td>
<td>285</td>
<td>237</td>
<td>22</td>
<td>74</td>
</tr>
<tr>
<td>1.234565e–20</td>
<td>208</td>
<td>341</td>
<td>335</td>
<td>23</td>
<td>80</td>
</tr>
</tbody>
</table>

(1) Gay (1990), 6 places
(2) Steele and White (1990), 6 places
(3) Gay (1990), maximum precision
(4) C library routine `ecvt`, 6 places
(5) C library function `sprintf("%g")`, 6 places

Maximum Precision: Uses

*Set membership*

Different numerical set objects always display differently

*Communication with solvers*

Equivalent text and binary forms are available

*Debugging results*

Exact results can be viewed
Unexpected rounding anomalies can be diagnosed . . .
Solution Precision

scheduling given demands and acceptable schedules

```
set SHIFTS;       # shifts
param Nsched;     # number of schedules;
set SCHEDULE = 1..Nsched; # set of schedules
set SHIFT_LIST (SCHEDULE) within SHIFTS; # shifts worked in each schedule
param required (SHIFTS) >= 0; # workers needed on each shift

minimize Total_Cost;
subject to Shift_Needs (i in SHIFTS): to_come >= required[i];
var Work (j in SCHEDULE) >= 0,
    obj Total_Cost 1, coeff (i in SHIFT_LIST[j]) Shift_Needs[i] 1;
```

```
set SHIFTS := Mon1 Tue1 Wed1 Thu1 Fri1 Sat1
             Mon2 Tue2 Wed2 Thu2 Fri2 Sat2
             Mon3 Tue3 Wed3 Thu3 Fri3 ;
param Nsched := 126 ;
set SHIFT_LIST[1] := Mon1 Tue1 Wed1 Thu1 Fri1 ;
set SHIFT_LIST[2] := Mon1 Tue1 Wed1 Thu1 Fri2 ;
set SHIFT_LIST[3] := Mon1 Tue1 Wed1 Thu1 Fri3 ; .....
```

Solution Precision (cont’d)

```
ampl: model sched.mod;
ampl: data sched.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
19 iterations, objective 265.6
ampl: option omit_zero_rows 1;
ampl: option display_eps .000001;
ampl: display Work;
Work [*] :=
  10 28.8 73 28
  10 7.6 87 14.4
  24 6.8 106 23.2
  30 14.4 109 14.4
  35 6.8 113 14.4
  66 35.6 123 35.6
  71 35.6
;
ampl: display sum (j in SCHEDULE) ceil(Work[j]); # objective rounded up
   sum{j in SCHEDULE} ceil(Work[j]) = 273
ampl: display 29+8+7+15+7+36+36+28+15+36+24+15+15+15+36;
   29 + 8 + 7 + 15 + 7 + 36 + 36 + 28 + 15 + 24 + 15 + 15 + 15 + 36 = 271
```
Solution Precision (cont’d)

```ampl
model sched.mod;
data sched.dat;
solve;
MINOS 5.5: optimal solution found.
19 iterations, objective 265.6
option display_precision 0;
display Work;
Work [*] :=
  10 28.799999999999997 73 28.000000000000018
  18 7.599999999999998 87 14.399999999999995
  24 6.799999999999999 95 -5.876671973951407e-15
  30 14.400000000000001 106 23.200000000000006
  35 6.799999999999999 108 4.685288280240683e-16
  55 -4.939614313857677e-15 109 14.4
  66 35.6                     113 14.4
  71 35.6 99999999999999994 123 35.59999999999999
```

Solution Precision (cont’d)

```ampl
option solution_round 6;
solve;
MINOS 5.5: optimal solution found.
19 iterations, objective 265.6
display Work;
Work [*] :=
  10 28.8 73 28
  18 7.6 87 14.4
  24 6.8 106 23.2
  30 14.4 109 14.4
  35 6.8 113 14.4
  66 35.6 123 35.6
  71 35.6
;
display sum {j in SCHEDS} ceil(Work[j]);
sum{j in SCHEDS} ceil(Work[j]) = 271
```
Other Precision and Rounding Options

```
AMPL: option *precision*;
option MD_precision 0;
option csvdisplay_precision 0;
option display_precision 6;
option expand_precision 6;
option objective_precision 10;
option output_precision 0;
option print_precision 0;
option solution_precision '';
AMPL: option *round*;
option csvdisplay_round '';
option display_round '';
option expand_round '';
option print_round '';
option solution_round '';```

Motivation

Treat all simple bounds the same, however declared

\[ \text{var Sell} \{ \text{p in PROD, t in } 1..T \} \geq 0, \leq \text{market}[p,t]; \]

\[ \text{var Sell} \{ \text{PROD,1..T} \} \geq 0; \]

\[ \text{subj to MLim} \{ \text{p in PROD, t in } 1..T \}: \text{Sell}[p,t] \leq \text{market}[p,t]; \]

Remove fixed variables, redundant constraints

Idea

Substitute bounds into constraints to deduce tighter bounds

Based on

Presolve 1: Variable Defined as Fixed

multi-period production planning

```
ampl: model steelT.mod;
ampl: data steelT.dat;
ampl: option show_stats 1;
ampl: solve;
Presolve eliminates 2 constraints and 2 variables.
Adjusted problem:
24 variables, all linear
12 constraints, all linear; 38 nonzeros
1 linear objective; 24 nonzeros.
MINOS 5.5: optimal solution found.
15 iterations, objective 515033
```

```
ampl: print {j in 1.._nvars: _var[j].status = "pre"}: _varname[j];
Inv['bands',0]
Inv['coils',0]
ampl: print {i in 1.._ncons: _con[i].status = "pre"}: _conname[i];
Init_Inv['bands']
Init_Inv['coils']
ampl: show Init_Inv;
subject to Init_Inv{p in PROD} : Inv[p,0] == inv0[p];
```

Presolve 2: Redundancy Implied by Original Bounds

diet cost minimization

```
ampl: model dietu.mod;
ampl: data dietu.dat;
ampl: solve;
Presolve eliminates 3 constraints.
Adjusted problem:
8 variables, all linear
5 constraints, all linear; 39 nonzeros
1 linear objective; 8 nonzeros.
MINOS 5.5: optimal solution found.
5 iterations, objective 74.27382022
```

```
ampl: print {i in 1.._ncons: _con[i].status = "pre"}: _conname[i];
Diet_Min['B1']
Diet_Min['B2']
Diet_Max['A']
```
Presolve 2 (cont’d)

```ampl
ampl: show Diet_Min;
subj to Diet_Min{i in MINREQ}: sum{j in FOOD} amt[i,j]*Buy[j] >= n_min[i];
```

```ampl
ampl: show Diet_Max;
subj to Diet_Max{i in MAXREQ}: sum{j in FOOD} amt[i,j]*Buy[j] <= n_max[i];
```

```ampl
ampl: show Buy;
var Buy{j in FOOD} >= f_min[j], <= f_max[j];
```

```ampl
ampl: display n_min;
n_min [*] :=
  A    700
  B1    0
  B2    0
  C    700
  CAL 16000;
```

```ampl
ampl: display {i in MAXREQ} (n_max[i], sum {j in FOOD} amt[i,j]*f_max[j]);
: n_max[i] sum{j in FOOD} amt[i,j]*f_max[j] :=
  A     20000                2860
  CAL   24000                34700
  NA    50000                91450;
```

Presolve 3: Redundancy Implied by Inferred Bounds
multi-commodity transportation

```ampl
ampl: model multi.mod;
ampl: data multi.dat;
ampl: solve;
```

Presolve eliminates 7 constraints and 3 variables.
Adjusted problem:
60 variables, all linear
44 constraints, all linear; 165 nonzeros
1 linear objective; 60 nonzeros.
MINOS 5.5: optimal solution found.
41 iterations, objective 199500

```ampl
ampl: print {j in 1.._nvars: _var.status[j] = "pre"}: _varname[j];
Trans['GARY','LAN','plate']
Trans['CLEV','LAN','plate']
Trans['PITT','LAN','plate']
```

```ampl
ampl: print {i in 1.._ncons: _con[i].status = "pre"}: _conname[i];
Demand['LAN','plate']
Multi['GARY','LAN']
Multi['GARY','WIN']
Multi['CLEV','LAN']
Multi['CLEV','WIN']
Multi['PITT','LAN']
Multi['PITT','WIN']
```
Presolve 3 (cont’d)

```ampl
ampl: expand Demand['LAN','plate'];
subject to Demand['LAN','plate']:
    Trans['GARY','LAN','plate'] + Trans['CLEV','LAN','plate'] + Trans['PITT','LAN','plate'] = 0;

ampl: show Multi;
subject to Multi {i in ORIG, j in DEST}:
    sum {p in PROD} Trans[i,j,p] <= limit[i,j];

ampl: display {i in ORIG, j in DEST} sum {p in PROD} Trans[i,j,p].ub - limit[i,j] [*,*] (tr)
    : CLEV GARY PITT :=
    DET Infinity Infinity Infinity
    FRA Infinity Infinity Infinity
    FRE Infinity Infinity Infinity
    LAF Infinity Infinity Infinity
    LAN Infinity Infinity Infinity
    STL Infinity Infinity Infinity
    WIN Infinity Infinity Infinity
;

ampl: show Supply;
subject to Supply {i in ORIG, p in PROD}:
    sum {j in DEST} Trans[i,j,p] == supply[i,p];

ampl: show Demand;
subject to Demand {j in DEST, p in PROD}:
    sum {i in ORIG} Trans[i,j,p] == demand[j,p];
```

Presolve 3 (cont’d)

```ampl
ampl: display {i in ORIG, j in DEST} sum {p in PROD} (Trans[i,j,p].ub2) - limit[i,j] [*,*] (tr)
    : CLEV GARY PITT :=
    DET 400 400 400
    FRA 275 275 275
    FRE 325 325 325
    LAF 375 375 375
    LAN -125 -125 -125
    STL 825 600 825
    WIN -250 -250 -250
;
```

20th Biennial Conference on Numerical Analysis
Dundee, Scotland, June 24-27, 2003
Presolve 4: Infeasibility

time-constrained production

```ampl
set PROD; # products
param rate (PROD) > 0;  # produced tons per hour
param avail >= 0;      # hours available in week
param profit (PROD);   # profit per ton
param commit (PROD) >= 0;  # lower limit on tons sold in week
param market (PROD) >= 0; # upper limit on tons sold in week

var Make {p in PROD} >= commit[p], <= market[p];
maximize Total_Profit : sum {p in PROD} profit[p] * Make[p];
subject to Time : sum {p in PROD} (1/rate[p]) * Make[p] <= avail;
```

```ampl
ampl: model steel3.mod;
ampl: data steel3.dat;
ampl: let avail := 13;
ampl: solve;
presolve: constraint Time cannot hold:
   body <= 13 cannot be >= 13.2589; difference = -0.258929
```

Presolve 4 (cont’d)

```ampl
ampl: display sum {p in PROD} (1/rate[p]) * Make[p].lb;
   sum{p in PROD} 1/rate[p] * (Make[p].lb) = 13.2589
ampl: let avail := 13.2589;
ampl: solve;
presolve: constraint Time cannot hold:
   body <= 13.2589 cannot be >= 13.2589; difference = -2.85714e-05
ampl: let avail := 13.25895;
ampl: solve;
MINOS 5.5: optimal solution found.
0 iterations, objective 61750.10714
ampl: let avail := 13.258925;
ampl: solve;
presolve: constraint Time cannot hold:
   body <= 13.2589 cannot be >= 13.2589; difference = -3.57143e-06
Setting $presolve_epsmax >= 4.29e-06 might help.
ampl: option presolve_epsmax;
option presolve_epsmax 1e-5;
```
Presolve 4 (cont’d)

```ampl
ampl: let avail := 13.258925;
ampl: solve;
presolve: constraint Time cannot hold:  
body <= 13.2589 cannot be >= 13.2589; difference = -3.57143e-06  
Setting $presolve_eps >= 4.29e-06 might help.
ampl: option presolve_eps;
option presolve_eps 0;
ampl: option presolve_eps 1e-5;
ampl: solve;
MINOS 5.5: optimal solution found.  
0 iterations, objective 61749.98214
ampl: option solver cplex;
ampl: solve;
CPLEX 8.1.0: Bound infeasibility column 'xl'.  
infeasible problem.
ampl: option cplex_options 'feasibility=5e-3';
ampl: solve;
CPLEX 8.1.0: optimal solution; objective 194828.5714  
1 dual simplex iterations (0 in phase I)
```

Works for Big LPs, Too!

```ampl
ampl: option show_stats 2;
ampl: model xxx1.mod;
ampl: data xxx1.dat;
ampl: solve;
Presolve eliminates 1769 constraints and 8747 variables.  
"option presolve 10;" used, but "option presolve 4;" would suffice.  
Adjusted problem:  
19369 variables, all linear  
3511 constraints, all linear; 150362 nonzeros  
1 linear objective; 19369 nonzeros.  
# 2 sec in AMPL,  
# 1/2 sec in presolve
ampl: reset;
ampl: model xxx2.mod;
ampl: data xxx2.dat;
ampl: solve;
Presolve eliminates 32989 constraints and 54819 variables.  
"option presolve 10;" used, but "option presolve 2;" would suffice.  
Adjusted problem:  
327710 variables, all linear  
105024 constraints, all linear; 1359068 nonzeros  
1 linear objective; 317339 nonzeros.  
# 23 sec in AMPL,  
# 4 sec in presolve
```
### Presolve Tolerances

**Bound fixing**
- `presolve_fixeps` 0
- `presolve_fixepsmax` 1e-5

**Constraint redundancy**
- `constraint_drop_tol` 0

**Bound infeasibility**
- `presolve_eps` 0
- `presolve_epsmax` 1e-5

**Integer bound rounding**
- `presolve_inteps` 1e-6
- `presolve_intepsmax` 1e-5

---

### Presolve Computations: Directed Rounding

**Principles**
- Round lower bounds toward $-\infty$
- Round upper bounds toward $+\infty$

**Practice**
- Fewer false alarms in presolve
  - can leave `presolve_eps` at 0
- Not performed properly by some computers & compilers
Influence on Solvers

Derivatives

Interaction of modeling language with nonlinear solvers
Automatic differentiation
Second derivatives and partial separability
Interior-point solvers

Complementarity problems

Forms of complementarity constraints
Solvers of square equilibrium problems and mathematical programs with equilibrium constraints

How AMPL Expresses a Nonlinear Problem

Just write nonlinear expressions

```plaintext
set J := 1 .. 18;
set K := 1 .. 7;
set PAIRS in {J,K};
set I := 1 .. 16;
param b {I} >= 0, default 0;
param c {PAIRS};
param E {PAIRS,I} integer;
param xlb >= 0, default 0;
var x {PAIRS} >= xlb, default 0.1;
minimize Energy:
  sum {(j,k) in PAIRS} x[j,k] * (c[j,k] + log (x[j,k] / sum {m in J:(m,k) in PAIRS} x[m,k]));
subject to H {i in I}:
  sum {(j,k) in PAIRS} E[j,k,i] * x[j,k] = b[i];
```

```
How AMPL Interacts with a Solver

User types...

```
option solver yrslv;
option yrslv_options "maxiter=10000";
solve;
```

AMPL...

Writes `at13151.nl`
Executes “`yrslv at13151 -AMPL`”

YRSLV “driver”...

Reads `at13151.nl`
Gets environment variable `yrslv_options`
Calls YRSLV routines to solve the problem
Writes `at13151.sol`

AMPL...

Reads `at13151.sol`

How a Driver Interacts with a Nonlinear Solver

Reads .nl problem file

- Loads everything into ASL data structure
- Copies linear coefficients, bounds, etc. to solver’s arrays
- Sets directives indicated by _options string

Runs algorithm

- Uses ASL data structure
  - to compute nonlinear expression and derivative values

Writes .sol solution file

- Generates result message
- Writes values of variables
- Writes other solution values, as appropriate
How the .nl File Represents a Nonlinear Problem

File contents
- Numbers of variables, constraints,
  integer variables, nonlinear constraints, etc.
- Coefficient lists for linear part
- Expression tree for nonlinear part plus sparsity pattern of derivatives

Expression tree nodes
- Variables, constants
- Binary, unary operators
- Summations
- Function calls
- Piecewise-linear terms
- If-then-else terms

```
  *             +
```


“Backward” Automatic Differentiation

Computations
- Forward sweep: compute \( \Phi(x) \),
  save info on \( \partial f(x) / \partial o \) for each operation \( o \)
- Backward sweep: recur to compute \( \nabla f(x) \)

Complexity
- Small multiple of time for \( f(x) \) alone
- Potentially large multiple of space

Advantages
- More accurate, efficient than finite differencing
- \( O(n) \) vs. \( O(n^2) \) for symbolic differentiation or forward AD
- Correct results for nondifferentiable functions
  (min, max, if-then-else, piecewise-linear)
2nd Derivatives

**Hessian-vector products:** $\nabla^2 f(x) \nu$

- Apply backward AD to compute gradients of $v^T \nabla f(x)$
- Equivalently, compute $\nabla_x (df(x + \tau \nu) / d\tau |_{\tau=0})$

**General case**

$\nabla^2 f(x) e_j$ for each $j = 1, \ldots, n$

**Partially separable case**

$f(x) = \sum_{i=1}^q f_i(U_i x)$ where $U_i$ is $m_i \times n$, $m \gg n$

$\nabla f(x) = \sum_{i=1}^q U_i^T \nabla f_i(U_i x)$

$\nabla^2 f(x) = \sum_{i=1}^q U_i^T \nabla^2 f_i(U_i x) U_i$, a sum of outer products

---

How AMPL Computes Hessians

**Detect partially separable structure**

- Walk expression tree
- Use a hashing scheme to spot common subexpressions
  
  ... may be useful even when Hessian is only approximated

**Compute derivative information**

- General or partially separable computations
- Dense or sparse
- Full or lower triangle
  
  ... using general and/or partially separable approach
Interior-Point Methods

To solve
Minimize \( f(x) \)
Subject to \( h_i(x) \geq 0, \ i = 1, \ldots, m \)

apply Newton’s method to the optimality conditions
\[
\nabla f(x) = \nabla h(x)^T \nu \\
h(x) = w \\
Wy = \mu e
\]

leading to a linear system of the form
\[
\begin{bmatrix}
-\nabla^2 \left( f(x) - h(x)^T \nu \right) & \nabla h(x)^T \\
\nabla h(x) & W^{-1}
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \nu
\end{bmatrix} = \cdots
\]

Developing Interior-Point Methods with AMPL

Write an AMPL driver
Use appropriate calls to get Hessian of Lagrangian

Convert some test problems
CUTE (734) COPS (17)
Schittkowski (195) Hock & Schittkowski (119)
Netlib (40) Vanderbei (29 groups)

... index at www.sor.princeton.edu/~rvdb/ampl/nlmodels/

Get some results
Rapid development of competitive interior-point solvers
— LOQO, KNITRO, MOSEK
Addition of 2nd-derivative options to other kinds of solvers
— CONOPT, SNOPT (?), PATHNLPl
Complementarity Problems

Definition
Collections of complementarity conditions:

- Two inequalities must hold,
  at least one of them with equality

Applications
Equilibrium problems in economics and engineering
Optimality conditions for nonlinear programs,
bi-level linear programs, bimatrix games, . . .

Classical Linear Complementarity

Economic equilibrium

| set PROD; # products         |
| set ACT; # activities        |
| param cost {ACT} > 0;        |
| param demand (PROD) >= 0;    |
| param io {PROD,ACT} >= 0;    |
| var Price {i in PROD};       |
| var Level {j in ACT};        |
| subject to Pri_Compl {i in PROD}: |
|     Price[i] >= 0 complements |
|     sum {j in ACT} io[i,j] * Level[j] >= demand[i]; |
| subject to Lev_Compl {j in ACT}: |
|     Level[j] >= 0 complements |
|     sum {i in PROD} Price[i] * io[i,j] <= cost[j]; |

... complementary slackness conditions
for an equivalent linear program
Mixed Linear Complementarity

*Economic equilibrium with bounded variables*

```plaintext
set PROD;   # products
set ACT;    # activities
param cost {ACT} > 0;       # cost per unit
param demand {PROD} >= 0;   # units of demand
param io {PROD,ACT} >= 0;   # units of product per unit of activity
param level_min {ACT} > 0;  # min allowed level for each activity
param level_max {ACT} > 0;  # max allowed level for each activity
var Price {i in PROD};
var Level {j in ACT};
subject to Pri_Compl {i in PROD}:
    Price[i] >= 0 complements
    sum {j in ACT} io[i,j] * Level[j] >= demand[i];
subject to Lev_Compl {j in ACT}:
    level_min[j] <= Level[j] <= level_max[j] complements
    cost[j] - sum {i in PROD} Price[i] * io[i,j];
```

... complementarity conditions
for optimality of an equivalent bounded linear program

Mixed Nonlinear Complementarity

*Economic equilibrium with price-dependent demands*

```plaintext
set PROD;   # products
set ACT;    # activities
param cost {ACT} > 0;       # cost per unit
param demand {PROD} >= 0;   # units of demand
param io {PROD,ACT} >= 0;   # units of product per unit of activity
param demzero {PROD} > 0;   # intercept and slope of the demand
param demrate {PROD} >= 0;  # as a function of price
var Price {i in PROD};
var Level {j in ACT};
subject to Pri_Compl {i in PROD}:
    Price[i] >= 0 complements
    sum {j in ACT} io[i,j] * Level[j] >= demzero[i] + demrate[i] * Price[i];
subject to Lev_Compl {j in ACT}:
    Level[j] >= 0 complements
    sum {i in PROD} Price[i] * io[i,j] <= cost[j];
```

... not equivalent to a linear program
Operands to complements: always 2 inequalities

Two single inequalities

\texttt{single-ineq1 complements single-ineq2}

Both inequalities must hold, at least one at equality

One double inequality

\texttt{double-ineq complements expr}

\texttt{expr complements double-ineq}

The double-inequality must hold, and

- if at lower limit then \texttt{expr} \geq 0
- if at upper limit then \texttt{expr} \leq 0
- if between limits then \texttt{expr} = 0

Influence on Solver Development

“Square” problems

\# of variables = \# of complementarity constraints + \# of equality constraints

Transformation to a simpler canonical form is possible

\textbf{MPECs}

Mathematical programs with equilibrium constraints

No restriction on numbers of variables & constraints

Objective functions permitted

\textbf{Consequences for developers}

People \textit{can} write MPECs in AMPL, so they \textit{do}

Demand for adapted solvers is increased

— interior (Vanderbei) & SQP (Fletcher & Leyffer) methods
Future Modeling Language Influences

Quadratic expressions
Automatic detection of quadratic terms & extraction of Hessian matrix
Original motivation? Convex quadratic objective, linear constraints

Matrix functions and constraints
Determinant, eigenvalues
Positive semidefinite
Original motivation? Semidefinite programming

Actual nonlinear expressions as input to solvers
Recursive walk of AMPL’s expression tree
Conversion to the form that the solver wants
Original motivation? Global optimization

AMPL Book 2nd Edition Now Available
### 2nd Edition Features

**New chapters**
- Database access
- Command scripts
- Modeling commands
- Interactions with solvers
- Display commands
- Complementarity problems

... all extensions previously only described roughly at web site

**Updates and improvements**
- Existing chapters extensively revised
- Updated reference manual provided as appendix

... and at half the recent price of the 1st edition!

... See [www.ampl.com/BOOK/](http://www.ampl.com/BOOK/)