An Optimized Overview of Optimization Software

Heather Gaddy, Heather Hardeman,
Jesse Patsolic, and Rebecca Waldrip

WAKE FOREST
UNIVERSITY

Department of Mathematics

11 April 2013

WAKE FOREST
UNIVERSITY
Department of Mathematics

Contents

Open Source Software

Definition

- Software that can be downloaded and used at no cost.
- Readily available code that can be read, redistributed, and modified by users.
- Software evolves as people improve upon, adapt, and fix bugs in code.

Examples of Open Source Software

- ADMB
- COIN-OR SYMPHONY
- IPOPT
- JOptimizer
- OpenSolver

OpenSolver

- An open source optimization program
- Designed to work with Microsoft Excel
- Compatible with built-in Excel solver, but more powerful
- Linear and integer programming optimizer

OpenSolver

- Developed and maintained by Andrew Mason and students from the Engineering Science department at the University of Auckland in New Zealand.
- Released under GNU General Public License.

Solving a Problem

- OpenSolver runs the open source COIN-OR CBC optimization engine.
- COIN-OR CBC is an open-source integer programming solver written in C++.
- Written by John J. Forrest and maintained by Ted Ralphs.

Solving a Problem

- Can be used alone or as the library code that runs another program, such as in the case of OpenSolver.
- OpenSolver analyzes Excel spreadsheet, builds model, passes it to the CBC engine.
- CBC then solves problem and result is read back into OpenSolver and loaded into the Excel spreadsheet.

QuickSolve Feature

- Often, user needs to solve a model after making only small changes to the model.
- Building a model can be tedious.
- QuickSolve allows small changes to be easily made in the model without having to reconstruct the entire model.

QuickSolve Feature (cont.)

- User can define the cells that will be changed as 'parameter' cells.
- User can then change the parameters and select QuickSolve to quickly solve the slightly modified model.

Other Advantages

- Includes a built-in model visualizer that highlights model's decision variables, objective functions, and constraints, making each piece clearly visible on a spreadsheet.
- No set limits on problem size.
- Useful in solve the linear relaxation of a non-linear model.

Potential Improvements

- A way to check that the objective function value and constraint values found by CBC match those calculated in the spreadsheet once optimal solution is entered in.
- Method to identify any non-linearities to the user.
- Benefit of open source software = individuals are free to work to try to improve the code in these ways.

Proprietary Software

Definition

Proprietary software is software that is owned by an individual or a company (usually the one that developed it). There are almost always major restrictions on its use, and its source code is almost always kept secret.

Examples of Proprietary Software

- LIONsolver
- Maple
- Mathematica
- Matlab
- OptimJ

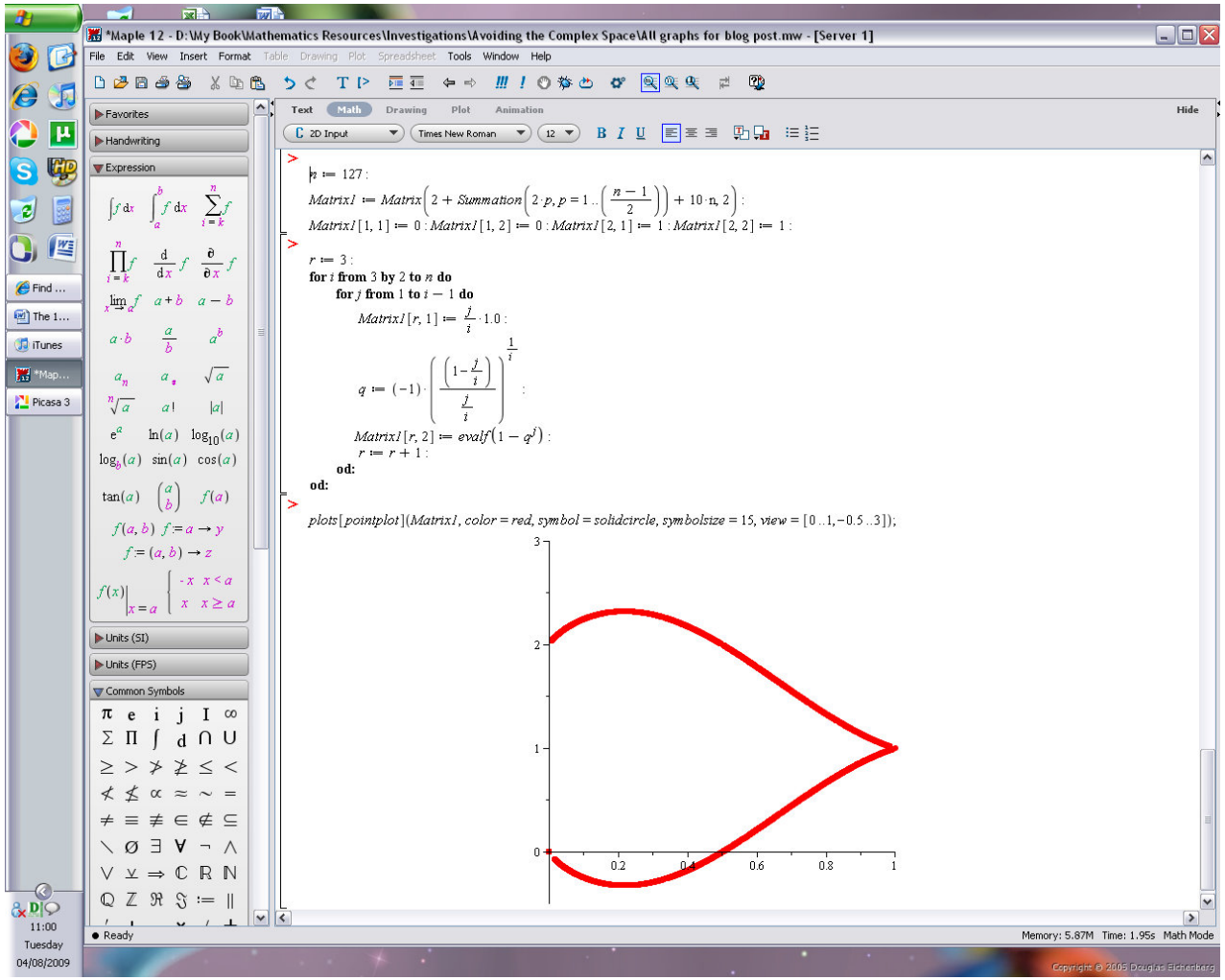

Maplesoft™
Mathematics • Modeling • Simulation
A Cybernet Group Company

How many languages do you speak?

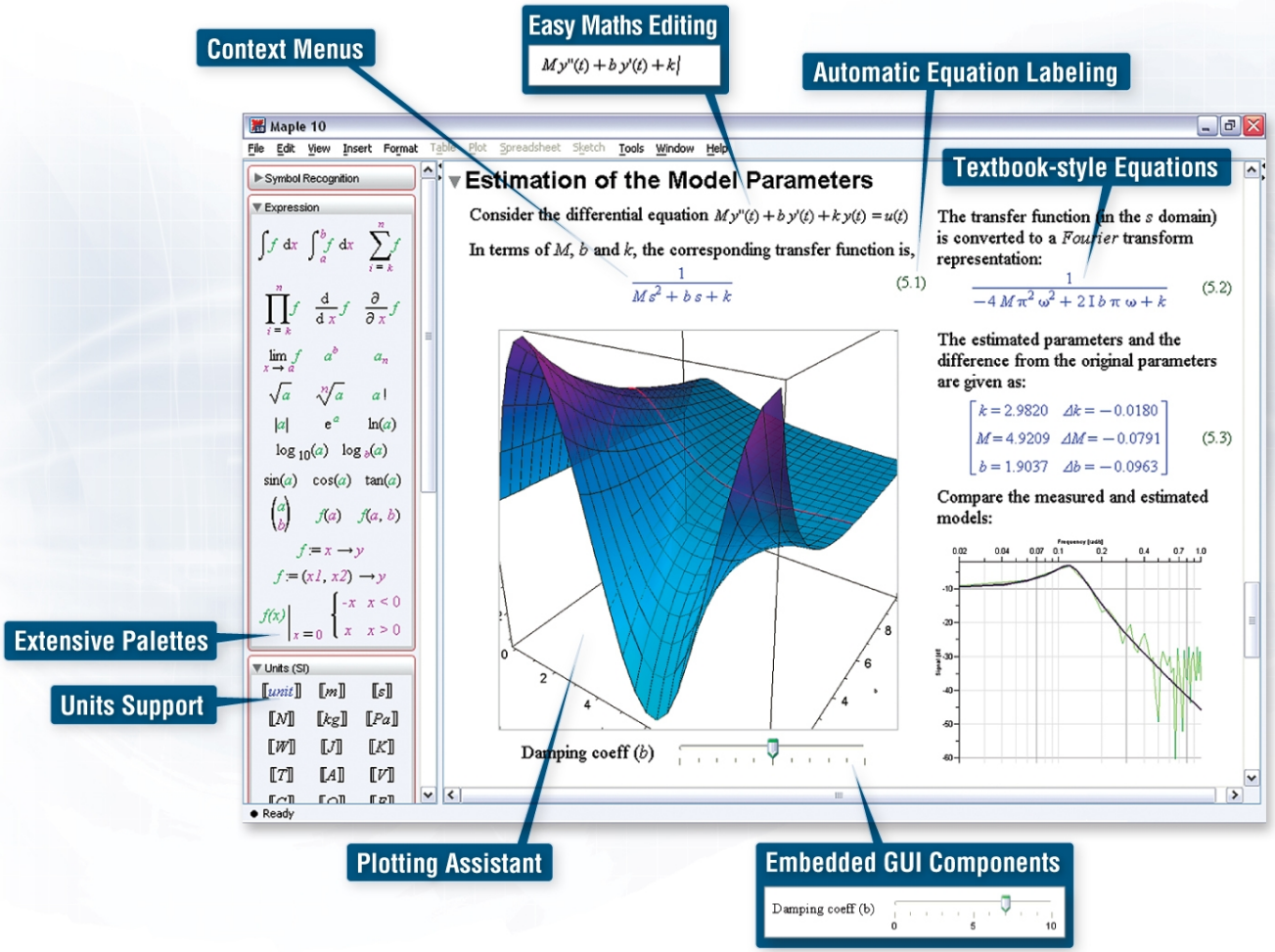
- C
- C#
- Java
- Matlab
- Visual Basic
- Excel

More Features

- Side panel to aid in typing functions
- Easy interface
- Handwritten symbol recognition
- Math equation editor
- Multiple packages and toolboxes



WAKE FOREST UNIVERSITY
Department of Mathematics



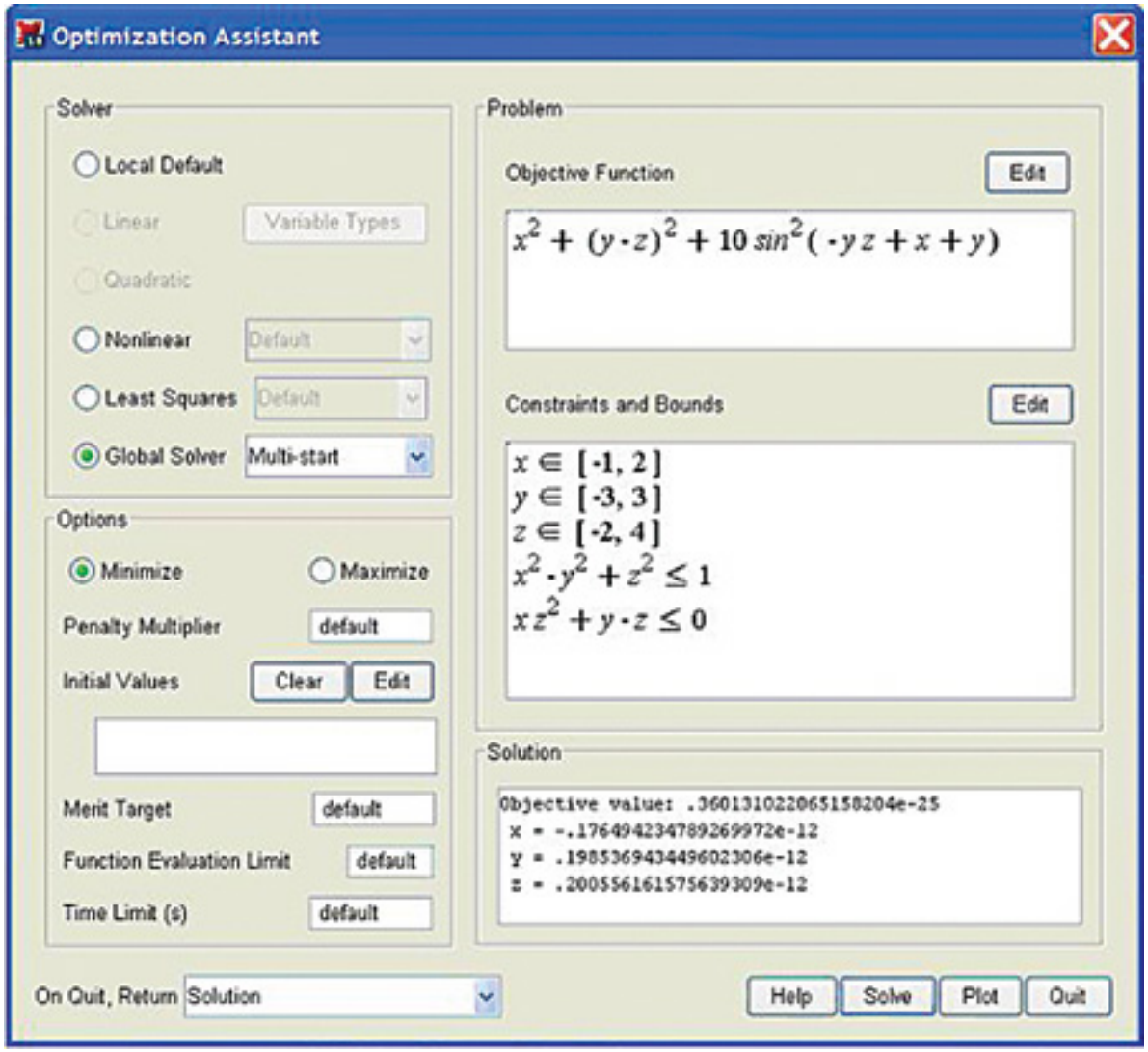
Wake Forest University
Department of Mathematics

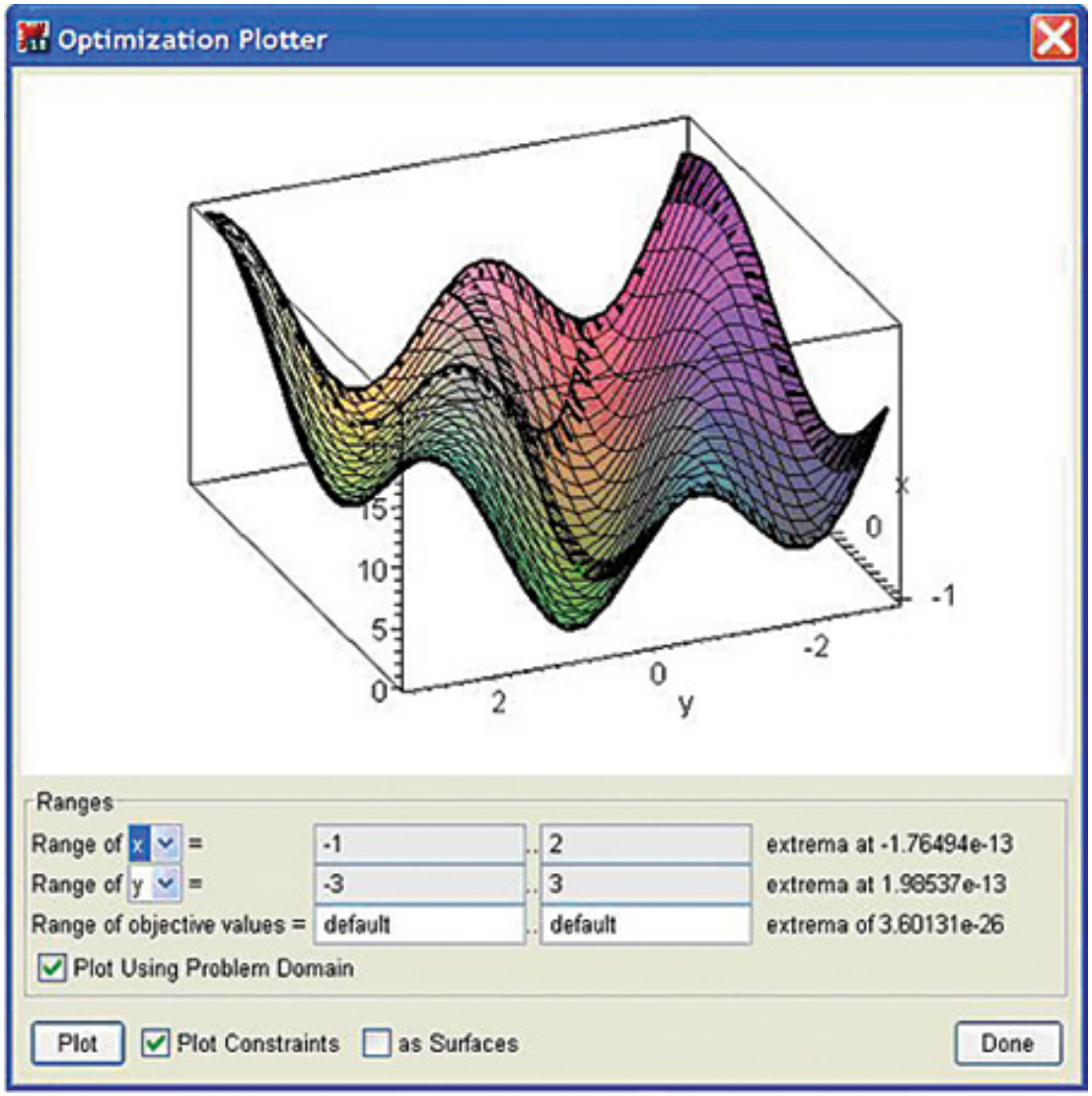
Functionality

- Math:
  - Integral Calculus
  - Solving Systems of Equations
  - Differential Equations (ODEs and PDEs)
  - Optimization
- Engineering
- Physics
- Statistical Modeling
- Financial Modeling
- Operations Research

Add-on Products and Apps

- Add-ons
  - Maple Global Optimization Toolbox
  - Maple Player (+ for iPad)
  - Teaching Calculus with Maple: A Complete Kit
- Apps
  - Interpolation and Smoothing
  - Grayscale Image Steganography
  - Diet Optimization


Optimization Assistant
Solver
Local Default
Linear
Variable Types
Quadratic
Nonlinear
Default
Least Squares
Default
Global Solver
Multi-start
Options
Minimize
Maximize
Penalty Multiplier
default
Initial Values
Clear
Edit
Merit Target
default
Function Evaluation Limit
default
Time Limit (s)
default
Problem
Objective Function
Edit
$x^2 + (y-z)^2 + 10\sin^2(-yz + x + y)$
Constraints and Bounds
Edit
$x \in [-1, 2]$
$y \in [-3, 3]$
$z \in [-2, 4]$
$x^2 - y^2 + z^2 \le 1$
$xz^2 + y - z \le 0$
Solution
Objective value: .36013102206515820 4e-25
x = -.17649423478926997 2e-12
y = .19853694344960230 6e-12
z = .20055616157563930 9e-12
On Quit, Return
Solution
Help
Solve
Plot
Quit


Optimization Plotter
15
10
5
0
2
0
y
-2
x
0
-1
Ranges
Range of x =
-1
2
extrema at -1.76494e-13
Range of y =
-3
3
extrema at 1.98537e-13
Range of objective values =
default
default
extrema of 3.60131e-26
Plot Using Problem Domain
Plot
Plot Constraints
as Surfaces
Done

Freeware

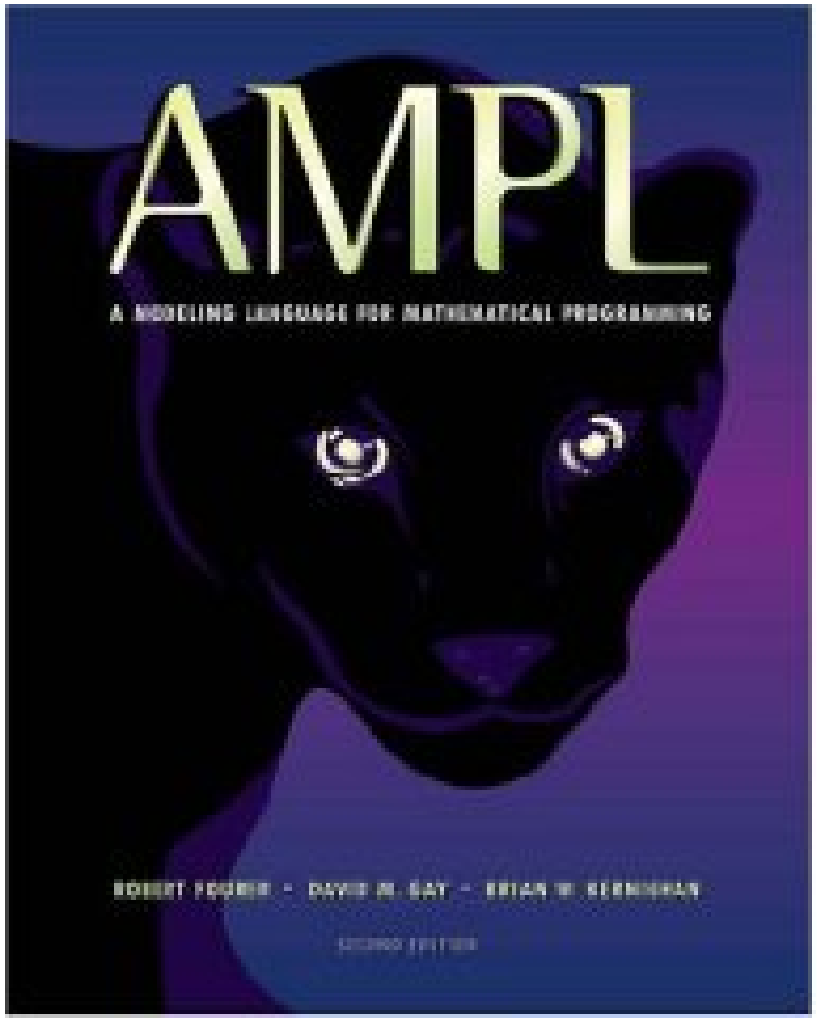
Definition

Freeware is software which has no monetary cost but is often more restricted than the professional version.

Examples of Optimization Freeware

- AIMMS
- AMPL
- Galahad library
- LIONsolver
- MINTO
- OptimJ
- SCIP

AMPL: A Mathematical Programming Language



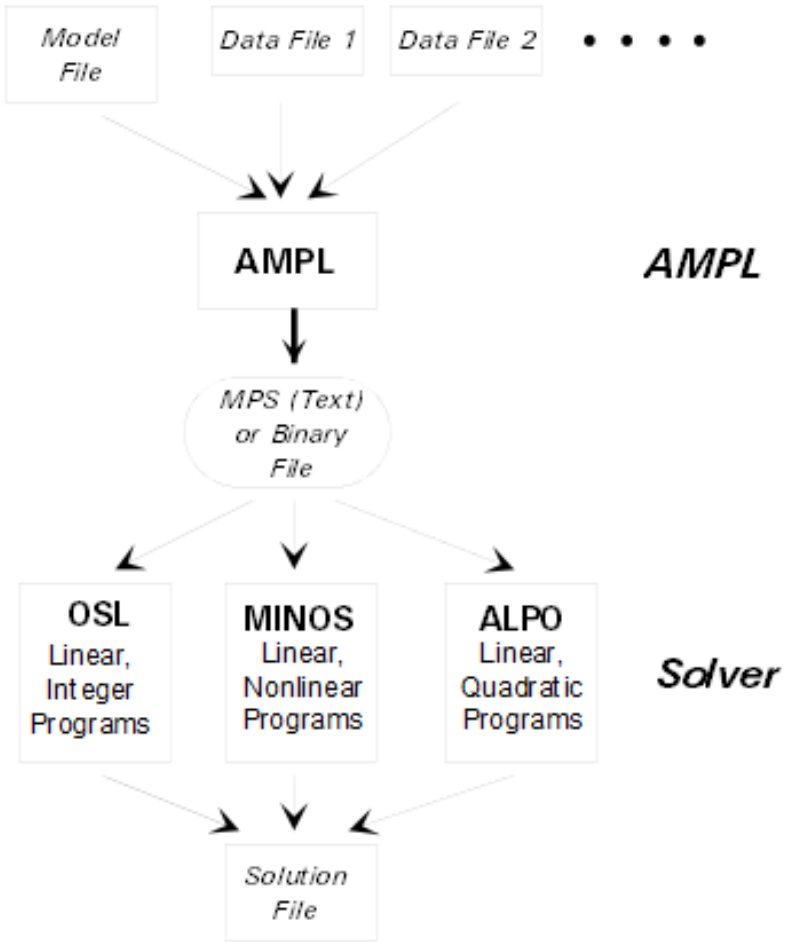
AMPL Language Features

- can work with a wide variety of sets and set operators
- syntax is organic and familiar
- Nonlinear programming features
- Convenient alternative notations

AMPL Environment Features

- interactive command environment
- new looping and if-then-else commands
- separation of model and data
- interfaces to a variety of solvers

How AMPL Works



Solvers

For linear programming:

- Continuous: BPMPD, CPLEX, LAMPS, LOQO, lp_solve, MINOS, MOSEK, OSL, SOPT, XA, Xpress-MP
- Integer: CPLEX, LAMPS, lp_solve, MINTO, MOSEK, OSL, SOPT, XA, Xpress-MP
- Network: CPLEX, OSL

Solvers

For nonlinear programming:

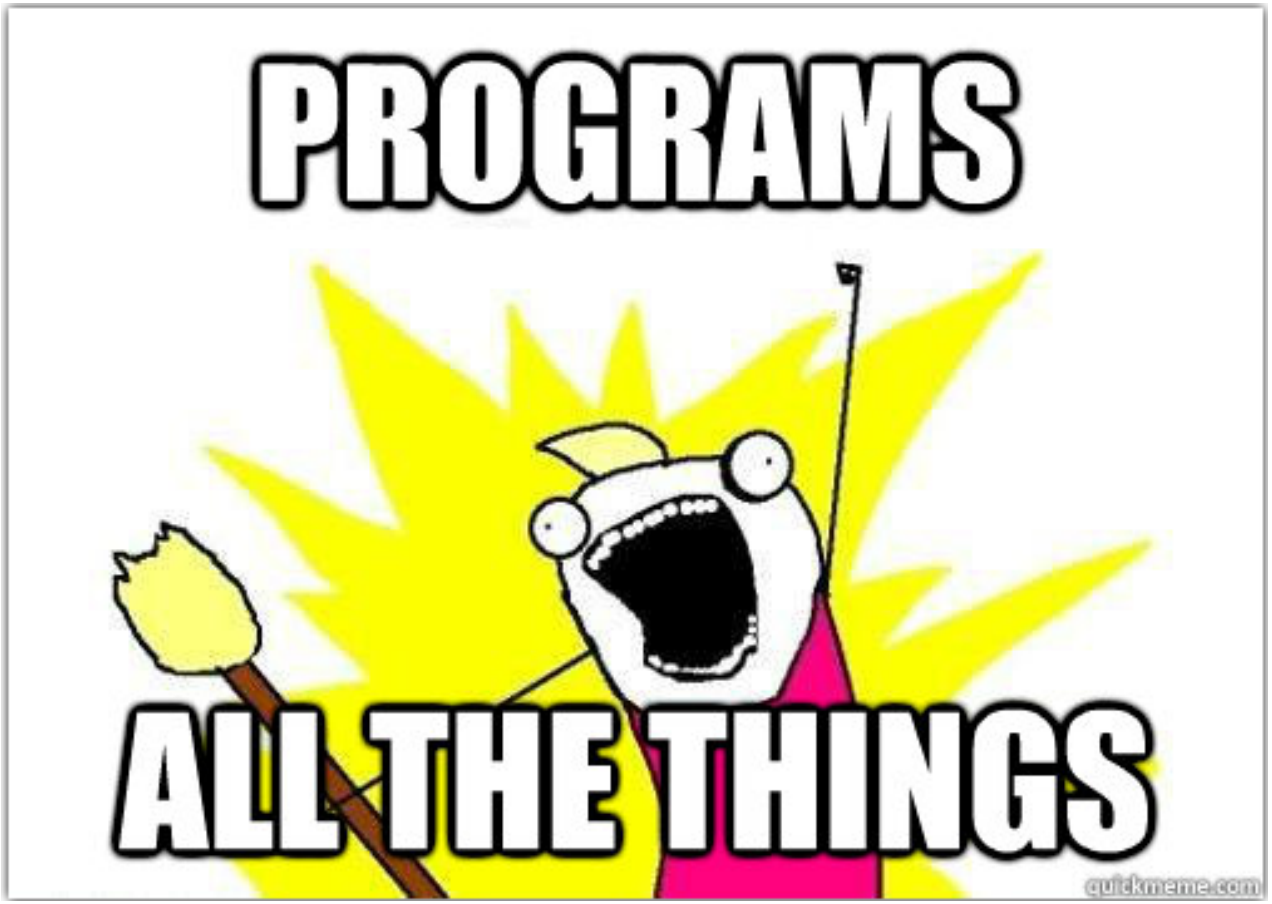
- Quadratic: CPLEX, MOSEK, OSL
- Convex: MOSEK, SOPT
- General continuous: CONOPT, DONLP2, FILTER, FSQP, IPOPT, KNITRO, LANCELOT, LOQO, MINOS, NPSOL, PENNON, SNOPT
- General integer: MINLP

Programming

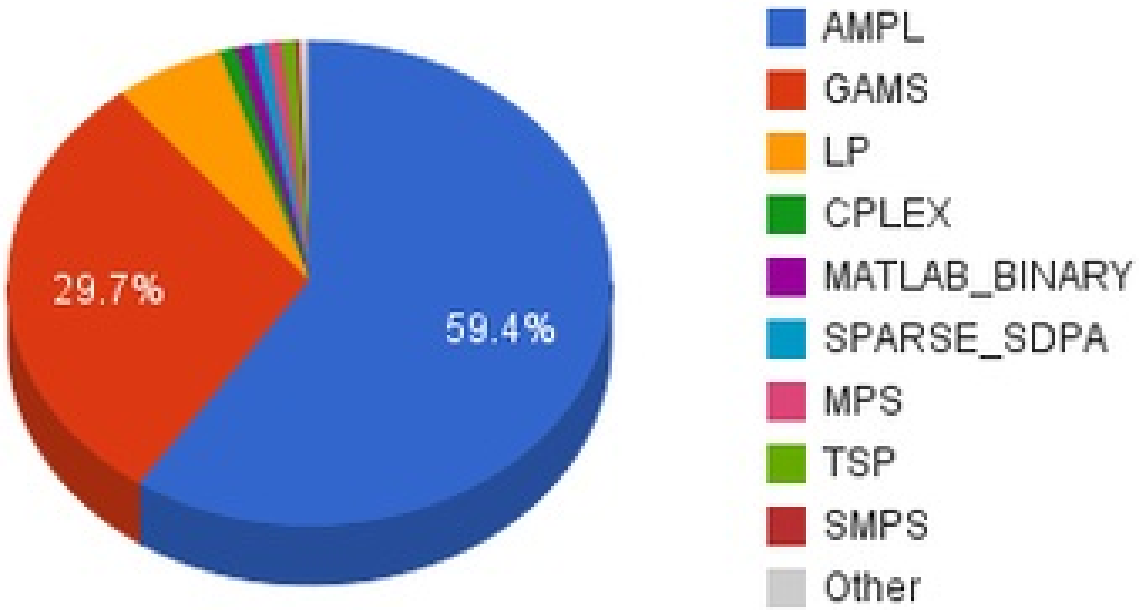
- Linear programming
- Quadratic programming
- Nonlinear programming
- Mixed-integer programming
- Mixed-integer quadratic programming with or without convex quadratic constraints

Programming

- Mixed-integer nonlinear programming
- Second-order cone programming
- Global optimization
- Semidefinite programming problems with bilinear matrix inequalities
- Complementarity problems in discrete or continuous variables
- Constraint programming


PROGRAMS
ALL THE THINGS
quickmeme.com

NEOS Statistics



Maple vs. AMPL

AMPL and Rosenbrock

roseShort.mod

```
# Rosenbrock function

var x{1..2};
var f1 = 100*(x[2] - x[1]^2)^2;
var f2 = (1 - x[1])^2;

minimize f:
    f1 + f2;

data;
var x:=
        1  100
        2  100;
```

Plain Text Tab Width: 4 Ln 1, Col 1 INS

AMPL Output

```
ampl: model rose100.mod;
ampl: solve;
MINOS 5.5: optimal solution found.
59 iterations, objective 3.219657877e-17
Nonlin evals: obj = 166, grad = 165.
ampl: display x;
x [*] :=
1  1
2  1;
```

```
ampl: model rose10.mod;
ampl: solve;
MINOS 5.5: optimal solution found.
34 iterations, objective 9.099872342e-19
Nonlin evals: obj = 87, grad = 86.
ampl: display x;
x [*] :=
1  1
2  1;
```

```
ampl: model rose0.mod;
ampl: solve;
MINOS 5.5: optimal solution found.
17 iterations, objective 9.247040366e-19
Nonlin evals: obj = 47, grad = 46.
ampl: display x;
x [*] :=
1  1
2  1;
```

```
ampl: model rose1_2.mod;
ampl: solve;
MINOS 5.5: optimal solution found.
8 iterations, objective 6.361071927e-19
Nonlin evals: obj = 24, grad = 23.
ampl: display x;
x [*] :=
1  1
2  1;
```

```
param:  FOOD:                      cost  f_min  f_max :=
  "CikFLA"                         3.88     .      .
  "ChefBoy"                        0.88     .      1
  "ChunkLightTuna"                 2.38     .      .
  "ThaiKitchen"                    1.98     .      .
  "MarieCalBeef"                   2.50     .      .
  "EasyMac"                        1.88     .      .
  "AngusBaconCheese"               6.19     .      .
  "QuarterPounderwCheese"          5.75     .      .
  "Big_Mac"                        5.89     .      .
  "Filet-O-Fish"                   5.45     .      .
  "Classic_Chicken"                6.19     .      .
  "Fries_small"                    1.00     .      .
  "Sausage_McMuffin"               3.99     .      .
  "2%_Lowfat Milk"                 0.60     .      3
  "Orange_Juice"                   1.00     .      . ;
```

```
/ param:  NUTR:   n_min  n_max :=
        Cal      2000      .
        Carbo     350    375
        Protein    55      .
        VitA      100      .
        VitC      100      .
        Calc      100      .
        Iron      100      . ;
```

```
param amt (tr):
                              Cal   Carbo Protein   VitA   VitC  Calc  Iron :=
  "CikFLA"                    440      42      30      2      2    15    15
  "ChefBoy"                   220      31       7      4      0     2    10
  "ChunkLightTuna"            120       0      13      0      0     0     4
  "ThaiKitchen"               450     100       7      8      2     2     8
  "MarieCalBeef"              370      50      21      2      8     4     8
  "EasyMac"                   188      31       7      4      0    10     0
  "QuarterPounderwCheese"     520      41      30     15      6    30    20
  "AngusBaconCheese"          790      63      45     10      4    30    35
  "Big_Mac"                   550      46      25      6      2    25    20
  "Filet-O-Fish"              390      39      15      2      0    15    10
  "Classic_Chicken"           510      55      24      4      6    15    20
  "Fries_small"               230      29       3      0      0     2     4
  "Sausage_McMuffin"          450      32      21      6      0    25    10
  "2%_Lowfat Milk"            112      12       7     10      3    26     0
  "Orange_Juice"              150      34       2      5    142    15     2 ;
```

Shopping list model

```
# From http://ampl.com/EXAMPLES/MCDONALDS/diet1.mod
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];
```

```
ampl: solve;
MINOS 5.5: ignoring integrality of 15 variables
MINOS 5.5: optimal solution found.
10 iterations, objective 26.38241406
Objective = Total_Cost
ampl: display Buy;
Buy [*] :=
                CikFLA  0
               ChefBoy  1
         ChunkLightTuna  0
            ThaiKitchen  0
           MarieCalBeef  0
                EasyMac  0
       AngusBaconCheese  1.048
QuarterPounderwCheese  2.2223
               Big_Mac  0
           Filet-O-Fish  0
        Classic_Chicken  0
            Fries_small  0
       Sausage_McMuffin  0
       '2%_Lowfat Milk'  3
           Orange_Juice  4.43712;
```

Figure : We have reached $(1, 1)$.