A Painless AMPL/PATH Tutorial Introduction to Solving Complementarity Problems with a Case Study from CIM-EARTH[*]

Sou-Cheng (Terrya) Choi[†]

`ampl-tutorial-11.tex`, v11, revised May 7, 2013

Abstract

AMPL is an expressive programming language for modeling and solving algebraic problems in optimization. The de facto AMPL solver of choice for solving complementarity problems is PATH, developed by Dirkse, Ferris, and Munson. This tutorial introduces basic elements in AMPL (version 20120629) and showcases sample programs for solving complementarity problems by using PATH (version 4.7.03). We feature a case study from CIM-EARTH, a framework for solving computable general equilibrium models with applications in climate change and economic policies.

[*]This work was supported in part by grants from the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract DE-AC02-06CH11357, and by the National Science Foundation grant SES-0951576 through the Center for Robust Decision Making on Climate and Energy Policy.

[†]Computation Institute, University of Chicago and Argonne National Laboratory (sctchoi@ci.uchicago.edu).

1 Introduction

This is a gentle introduction to the well-known mathematical-optimization language AMPL [1] and the AMPL interface of the PATH solver [13]. We cover a handful of AMPL/PATH features in this tutorial, but the content is nevertheless sufficient for solving large-scale complementarity problems (CPs) with up to $O(10^5)$ variables. Readers can expect to substantially increase their productivity in solving CPs with AMPL/PATH.

The key references for AMPL/PATH are Chapter 19 of [8], and two articles by Ferris et al. [6, 7]. We take this opportunity to review and update the syntax features in AMPL and PATH that have evolved since the aforementioned publications appeared and present sample illustrative programs for solving CPs with AMPL version 20120629 and PATH version 4.7.03.

We start with a quick overview of AMPL before we delve into the language syntax. AMPL is a scripting language. We do not need to compile AMPL programs. It is a powerful commercial software product that specializes in modeling and solving a wide range of mathematical programming problems. A free "student" edition is available for download and installation by anyone, not just students, to learn how to use AMPL to solve the most common optimization problems containing up to a few hundred variables, objectives, and constraints. The software has many optimization solvers developed by various experts from both academia and industry. The default solver is MINOS, developed by Murtagh and Saunders [12]. Other commonly used solvers include CPLEX from IBM [3] and KNITRO from Ziena [10]. For complementarity problems, the de facto solver in AMPL is PATH.

1.1 Hello World!

We follow the convention of starting a programming tutorial with "Hello World!". To print a literal string with the value "Hello World!" and a newline character ("`\n`") in AMPL, we can use the built-in function `printf`; its syntax is similar to the `printf` function in ANSI C except that there are no open and close parentheses around the string.

```
printf "Hello World!\n";
```

The double quotation marks around both ends of the string can be single quotation marks. We can also break up the string into shorter strings and call `printf` multiple times.

```
printf 'Hello ';
printf 'World!';
printf '\n';
```

Yet another way is to use the format string "`%s\n`" as the second argument of the `printf` function.

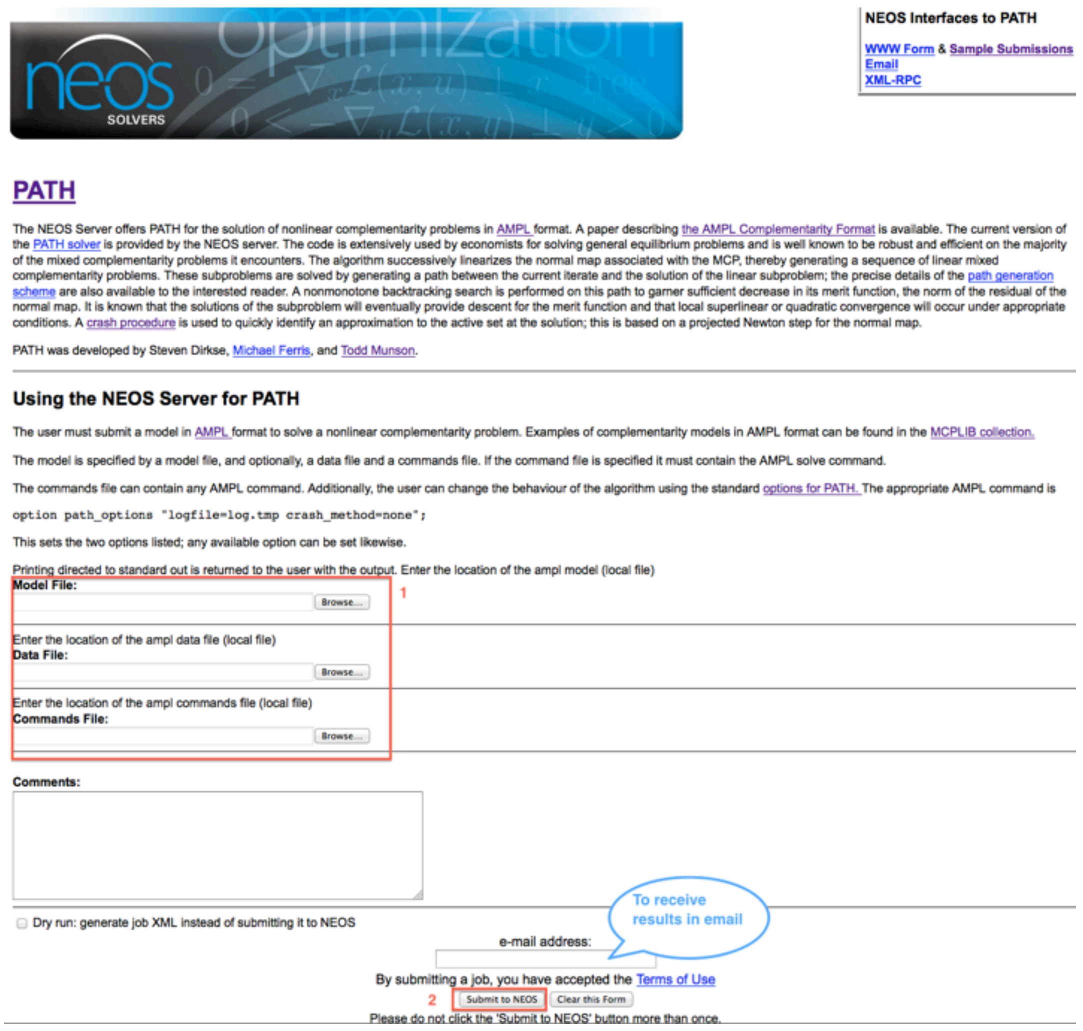
```
printf "%s\n", "Hello World!";
```

1.2 Running AMPL commands and programs

AMPL programs can be run in multiple ways. First, to run AMPL/PATH on the NEOS server [14], we can upload data and model files (Figure 1) using the NEOS/PATH web interface [13]. If in addition we provide an email address in the web form, then we will receive the computed results when the job is complete. The NEOS service offers great convenience because a user does not need to install, manage, or administer the application server or the software.

On a machine where AMPL is installed, a user may download a graphical user interface (GUI) suitable for the machine platform; see Figure 2. This feature is, however, marked "experimental" by AMPL.

Our preferred way is to run AMPL in a simple command-line interface. Suppose we have saved our first "Hello World!" program in a text file with the filename "`HelloWorld.ampl`". Then at an operating system (OS) prompt (indicated by "`$`"), we can simply run "`ampl HelloWorld.ampl`":



NEOS Interfaces to PATH

WWW Form & Sample Submissions
Email
XML-RPC

PATH

The NEOS Server offers PATH for the solution of nonlinear complementarity problems in AMPL format. A paper describing the AMPL Complementarity Format is available. The current version of the PATH solver is provided by the NEOS server. The code is extensively used by economists for solving general equilibrium problems and is well known to be robust and efficient on the majority of the mixed complementarity problems it encounters. The algorithm successively linearizes the normal map associated with the MCP, thereby generating a sequence of linear mixed complementarity problems. These subproblems are solved by generating a path between the current iterate and the solution of the linear subproblem; the precise details of the path generation scheme are also available to the interested reader. A nonmonotone backtracking search is performed on this path to garner sufficient decrease in its merit function, the norm of the residual of the normal map. It is known that the solutions of the subproblem will eventually provide descent for the merit function and that local superlinear or quadratic convergence will occur under appropriate conditions. A crash procedure is used to quickly identify an approximation to the active set at the solution; this is based on a projected Newton step for the normal map.

PATH was developed by Steven Dirkse, Michael Ferris, and Todd Munson.

Using the NEOS Server for PATH

The user must submit a model in AMPL format to solve a nonlinear complementarity problem. Examples of complementarity models in AMPL format can be found in the MCPLIB collection.

The model is specified by a model file, and optionally, a data file and a commands file. If the command file is specified it must contain the AMPL solve command.

The commands file can contain any AMPL command. Additionally, the user can change the behaviour of the algorithm using the standard options for PATH. The appropriate AMPL command is

```
option path_options "logfile=log.tmp crash_method=none";
```

This sets the two options listed; any available option can be set likewise.

Printing directed to standard out is returned to the user with the output. Enter the location of the ampl model (local file)
Model File:

Enter the location of the ampl data file (local file)
Data File:

Enter the location of the ampl commands file (local file)
Commands File:

Comments:

Dry run: generate job XML instead of submitting it to NEOS

e-mail address:

By submitting a job, you have accepted the Terms of Use

Please do not click the 'Submit to NEOS' button more than once.

Figure 1: Running AMPL/PATH on NEOS server.

```
$ ampl HelloWorld.ampl
```

Or we can run “**ampl**” at the OS prompt to start the command interface and execute our Hello World **printf** statement at the AMPL prompt “**ampl:**” as follows:

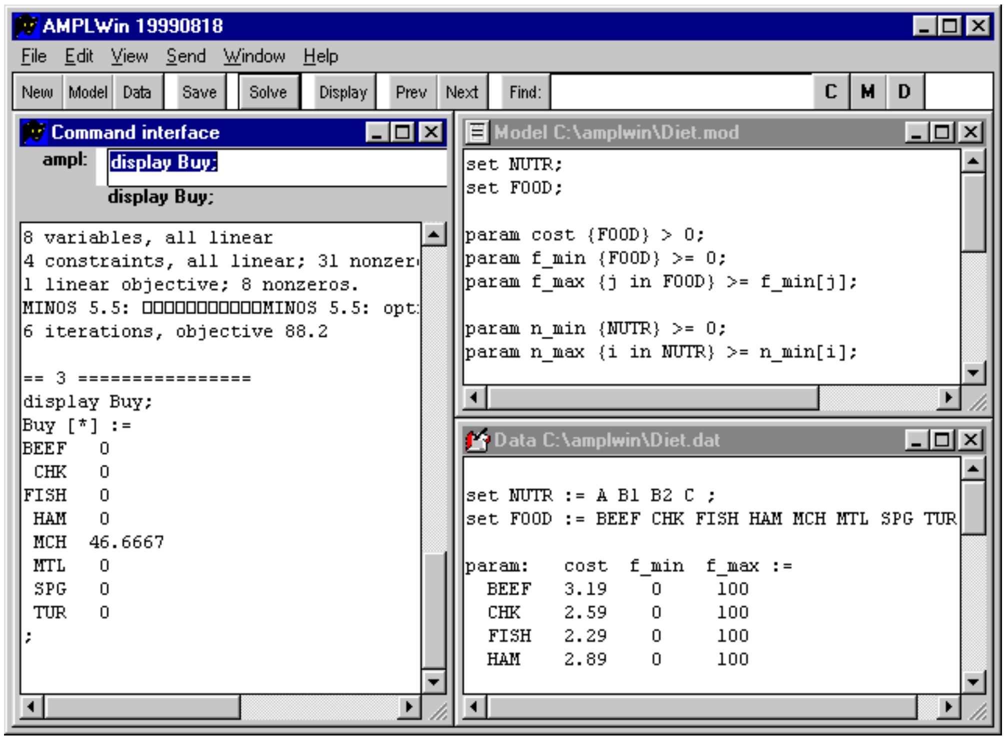
```
$ ampl
ampl: printf "Hello World!\n";
```

A third way is to issue “model [filename]” to execute AMPL commands saved in a text file.

```
ampl: model HelloWorld.ampl;
```

The first rule of AMPL is that every statement should finish with a semicolon (space is, however, unnecessary). Otherwise the user will be questioned by AMPL:

```
ampl: printf "Hello World!\n"
ampl?
```


AMPLWin 19990818
File Edit View Send Window Help
New Model Data Save Solve Display Prev Next Find:
C M D
Command interface
ampl:
Model C:\amplwin\Diet.mod
Data C:\amplwin\Diet.dat

```
display Buy;
display Buy;
```

```
8 variables, all linear
4 constraints, all linear; 31 nonzer
1 linear objective; 8 nonzeros.
MINOS 5.5: □□□□□□□□□□□□MINOS 5.5: opt
6 iterations, objective 88.2

== 3 ================
display Buy;
Buy [*] :=
BEEF    0
 CHK    0
FISH    0
 HAM    0
 MCH   46.6667
 MTL    0
 SPG    0
 TUR    0
;
```

```
set NUTR;
set FOOD;

param cost {FOOD} > 0;
param f_min {FOOD} >= 0;
param f_max {j in FOOD} >= f_min[j];

param n_min {NUTR} >= 0;
param n_max {i in NUTR} >= n_min[i];
```

```

set NUTR := A B1 B2 C ;
set FOOD := BEEF CHK FISH HAM MCH MTL SPG TUR

param:   cost  f_min  f_max :=
  BEEF   3.19    0     100
  CHK    2.59    0     100
  FISH   2.29    0     100
  HAM    2.89    0     100
```

Figure 2: Running AMPL using a GUI. Image credit: http://www.ampl.com/GUI.

2 Warming up

In this section, through a few easy AMPL programs, we highlight some of the frequently used tokens in the language:

- constant (`N`),
- variables (`i, j, p, x, y`),
- keywords (`param, var, let, in, minimize, subject to`),
- operators (`:=, *, ..`), expressions (e.g., `i * j`), statements (`;`),
- comments (`#`),
- control (`for`), and
- commands (`solve, printf, display`).

2.1 Printing a multiplication table

Our second AMPL program prints a multiplication table. The AMPL file is named `Mult.ampl`. At the OS prompt, we simply run "`ampl Mult.ampl`" to output the multiplication table, as shown in the left of Figure 3. The content of the AMPL program is listed in the right of Figure 3. It demonstrates a few basic but important AMPL features. Let us go through the program line by line. First is a comment that starts with a hex sign ("`#`") and ends at the end of the line. Then we declare a constant $N = 9$ and a variable p that is zero by default. After that, we have two nested "`for`" loops with integer indices i and j that run from 1 to N. Within the loops we compute the products of i and j, and store the results in the variable p before printing the value of p using `printf` with the format string "`%2d `" that reserves two spaces for each number and a blank between two consecutive integers.

```
$ ampl Mult.ampl
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81
```

```
# File: Mult.ampl
param N := 9;
var p;         # default 0

for {i in 1..N} {
  for {j in 1..N} {
    let p := i * j;
    printf "%2d ", p;
   }
   printf "\n";
}
```

Figure 3: Printing a multiplication table. Left: AMPL outputs. Right: AMPL code.

2.2 Solving linear system of equations

Our next program is a toy example that solves a system of nonsingular symmetric[1] linear equations in only two variables by casting it as a convex optimization problem, that is, minimizing its quadratic form; see the left part in Figure 4. The AMPL code in Quad2.ampl returns the unique solution $(2, 0)$ of the problem. We can see how concise it is to specify the problem in AMPL. We first declare two variables x and y. Then we use the AMPL keyword "`minimize`" followed by a objective-function label "`f:`" and its expanded quadratic form. At this point the convex problem is specified. The "`solve`" statement invokes the default optimization solver MINOS. We use AMPL's "`display`" command to print out the solution values of the variables $x = 2$ and $y = 0$. We note that AMPL always requires a label for every objective or constraint in the specification of an optimization problem.

Want to solve $Az = b$

$A \equiv \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad b \equiv \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

$z \equiv \begin{bmatrix} x \\ y \end{bmatrix}$

$$\Leftrightarrow \boxed{\min_{z \in \mathbb{R}^2} \frac{1}{2} z^T A z - b^T z}$$

```
$ ampl Quad2.ampl
x = 2
y = 0
```

```
# File: Quad2.ampl
var x;
var y;

minimize f:
  1/2 * (x^2 + 2 * x * y - y^2)
  - 2 * x - 2 * y ;

solve;
display x, y;
```

Figure 4: Solving a symmetric system of linear equations by minimizing its quadratic form.

Another way[2] to solve the linear system is to minimize a constant function with the linear equations as constraints. The meat of the AMPL code in this approach is listed in Figure 5. We use "`minimize f`" equal to one, or any constant number; "`subject to`" the first constraint, which is labeled by "`c1:`" and defined by the first equation; "`subject to`" the second constraint, labeled by "`c2`" and defined by the second equation. Once again, running the code in `Eqn2.ampl` gives the correct solutions.

[1] If A is unsymmetric ($A \neq A^T$), we can minimize the quadratic form of $\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$.

[2] This technique is directly applicable to unsymmetric systems.

Want to solve $Az = b$

$A \equiv \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$

$b \equiv \left[\begin{smallmatrix} 2 \\ 2 \end{smallmatrix}\right]$

$z \equiv \left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$

$\Leftrightarrow \boxed{\min_{z \in \mathbb{R}^2} \; 1 \text{ subject to } Az = b}$

```
$ ampl Eqn2.ampl
x = 2
y = 0
```

```
# File: Eqn2.ampl
var x;
var y;

minimize f:
  1;
subject to c1:
  x + y = 2;
subject to c2:
  x - y = 2;

solve;
display x, y;
```

Figure 5: Solving a system of linear equations by minimizing a constant function subject to the linear system.

3 Solving a CP with AMPL/PATH

At this point, we are ready to take on inequalities and complementarity problems. We first abstract the problem definitions in this section, and then we introduce the PATH solver.

3.1 Nonlinear complementarity problems

We formally define the scalar and general forms of nonlinear complementarity problems (NCPs) below:

- Given $f : \mathbb{R} \to \mathbb{R}$ continuously differentiable, solving a scalar NCP is equivalent to finding $x \in \mathbb{R}$ such that $\boxed{0 \le x} \quad \perp \quad \boxed{f(x) \ge 0}$, where the symbol "$\perp$" means equality must hold at one or both of its sides, i.e.,
$$\text{or} \begin{cases} x = 0 & \text{and} \quad f(x) \ge 0 \\ x > 0 & \text{and} \quad f(x) = 0 \end{cases}.$$
- Given $F : \mathbb{R}^n \to \mathbb{R}^n$ continuously differentiable, solving an NCP is equivalent to finding $x \in \mathbb{R}^n$ such that $\boxed{0 \le x} \quad \perp \quad \boxed{F(x) \ge 0}$, where the symbol "$\perp$" means $\boxed{0 \le x_i} \quad \perp \quad \boxed{F_i(x) \ge 0}$ for $i = 1, \ldots, n$.

To summarize, in the scalar case, we are given a *smooth* (i.e., continuously differentiable) scalar-valued function. A nonnegative x is a solution to the problem if one or both of the equalities are true at x. We call $x = 0$ a degenerate or trivial solution if $f(0) = 0$, that is, if both equalities hold. A general NCP is similar except that we have a vector-valued function. Now a vector x is a solution if it satisfies one or both of the inequalities *componentwise*. Clearly, if $n = 1$, then an NCP is a scalar problem.

An NCP is often associated with a system of nonlinear equations with nonnegative variables, but more often it is a mix of nonlinear equations and nonlinear inequities. If the vector-valued function $F(x)$ equals $Ax + b$ for some given square matrix or linear operator A of order n and some n-vector b, then it is reduced to a linear CP.

The following is an example of a scalar NCP with $f(x)$ being a quadratic function.

Example 3.1. *Find $x \in \mathbb{R}$ such that* $\boxed{0 \le x \;\perp\; f(x) \equiv (x-1)^2 - 1 \ge 0}$. *We consider the two cases $x = 0$ and $x > 0$:*
$$\textit{or} \begin{cases} x = 0 & \textit{and} \quad (x-1)^2 \ge 1 \quad \Rightarrow \quad x = 0 \text{ (degenerate)} \\ x > 0 & \textit{and} \quad (x-1)^2 = 1 \quad \Rightarrow \quad \boxed{x = 2} \end{cases}.$$
Obviously $x = 0$ is a degenerate solution since $f(x)$ is also zero. Thus $x = 2$ is the unique nontrivial solution.

3.2 Mixed complementarity problems

A more general class of CPs is mixed complementarity problems (MCPs). We formally define the scalar and general forms of MCPs below:

- Given $l, u \in \mathbb{R}$ and smooth $f : \mathbb{R} \to \mathbb{R}$, solving a scalar MCP is equivalent to finding $x \in \mathbb{R}$ such that $\boxed{l \le x \le u} \quad \perp \quad \boxed{f(x)}$, where the symbol "$\perp$" means that one of the following holds:

$$\text{or} \begin{cases} x = l & \text{and} \quad f(x) \ge 0 \\ x = u & \text{and} \quad f(x) \le 0 \\ l < x < u & \text{and} \quad f(x) = 0 \end{cases}.$$

- Given $l, u \in \mathbb{R}^n$ and smooth $F : \mathbb{R}^n \to \mathbb{R}^n$, solving an MCP is equivalent to finding $x \in \mathbb{R}^n$ such that $\boxed{l \le x \le u} \perp \boxed{F(x)}$, where $\perp$ means $\boxed{l_i \le x_i \le u_i} \perp \boxed{F_i(x)}$ for $i = 1, \ldots, n$.

To summarize, in the scalar case, we are given a lower bound l, an upper bound u, and a smooth scalar function. We want to find x in the open interval (l, u) such that one or more of the three defining cases are true. In the general form, an MCP works with vector bounds l and u, a smooth vector-valued function, so that at each component, we have a scalar MCP.

An NCP is actually a special case of an MCP with $l = 0$ and $u = +\infty$. If *both* bounds do not exist ($l = -\infty$ and $u = +\infty$), then we have a system of nonlinear equations; otherwise an MCP is a combination of nonlinear equalities and (double) inequalities. If $F(x) = Ax + b$ for some square matrix or linear operator A of order n, and some n-vector b, then we have a linear MCP.

The following is an instance of a scalar MCP. The scalar function is the same quadratic function as in Example 3.1, but we have an additional upper bound $u = 3$.

Example 3.2. *Find $x \in \mathbb{R}$ such that $\boxed{0 \le x \le 3 \quad \perp \quad f(x) \equiv (x-1)^2 - 1}$. We consider the three defining cases in turn:*

$$\text{or} \begin{cases} x = 0 & \text{and} \quad (x-1)^2 \ge 1 & \Rightarrow \quad x = 0 \text{ (degenerate)} \\ x = 3 & \text{and} \quad (x-1)^2 \le 1 & \Rightarrow \quad \text{※ (no solution)} \\ 0 < x < 3 & \text{and} \quad (x-1)^2 = 1 & \Rightarrow \quad \boxed{x = 2} \end{cases}.$$

The first case yields $x = 0$ as a degenerate solution since $f(x) = 0$. The second case has no solution since the RHS inequality is not satisfied at $x = 3$. The last case with $x \in (0, 3)$ and $f(x) = 0$ gives $x = 2$ as the unique nontrivial solution.

3.3 PATH solver for MCPs

For solving an MCP or an NCP, the de facto solver in AMPL is PATH. It is an iterative method that solves a linear MCP one step at a time. The solver has desirable convergence properties, being globally convergent to a local solution, and a locally quadratic convergence rate, which holds under some strong regularity assumptions. It has an "optional"[3] but powerful "native" preprocessor designed and written specifically for preprocessing input data, simplifying and transforming complementarity conditions. Implementations are done mostly in C, with some parts done in FORTRAN (LUSOL [9, 11]) and C++ (some other factorizations), with programming interfaces available in AMPL, as well as MATLAB and GAMS. Binary executable files of PATH for a number of platforms are available for free download [?].

Our first AMPL/PATH program solves the NCP in Example 3.1. Since the default AMPL solver is MINOS, we have to specify "`option solver pathampl`" to choose PATH as the solver. Then we declare and initialize x to a positive constant. Next we write down the label "`cond1:`", which is arbitrary but necessary in AMPL/PATH for the complementarity condition. Then we specify the inequalities separated by the AMPL keyword "complements". Calling "solve" invokes PATH, which returns the solution $x = 2$.

If we had initialized x to zero (i.e., `var x := 0`), AMPL/PATH would have returned the degenerate solution $x = 0$. Thus, in solving a CP in AMPL/PATH, it is always advisable to initialize variables to nonzeros.

[3]We strongly recommend enabling the PATH preprocessor.

Find $x \in \mathbb{R}$ such that

$$\boxed{0 \le x \ \perp\ f(x) \equiv (x-1)^2 - 1 \ge 0}$$

```
$ ampl NCP1.ampl
x = 2
```

```
# File: NCP1.ampl
option solver pathampl;
var x := 10;

cond1:
 0 <= x complements (x-1)^2 >= 1;

solve;
display x;
```

Figure 6: Solving an NCP in one variable using AMPL/PATH.

Our second AMPL/PATH program solves the scalar MCP in Example 3.2. The AMPL program is similar to that in Figure 6 except that we have double inequalities being orthogonal to the function definition.

Find $x \in \mathbb{R}$ such that

$$\boxed{0 \le x \le 3 \quad \perp \quad f(x) \equiv (x-1)^2 - 1}$$

```
$ ampl MCP1.ampl
x = 2
```

```
# File: MCP1.ampl
option solver pathampl;
var x := 10;

cond1:
 0 <= x <= 3 complements (x-1)^2 - 1;

solve;
display x;
```

Figure 7: Solving an MCP in one variable using AMPL/PATH.

4 Casting optimization problems as CPs

In this section we show two canonical optimization problems that often appear in economic applications, and we describe how to transform them into complementarity problems.

4.1 KKT to NCP

In broad strokes (see [15, Chapter 12] for details), if we have a nonlinear optimization problem with a smooth objective and smooth inequality constraints, then a local optimum satisfies the KKT conditions. In other words, given $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ continuously differentiable, if x^* is a local minimum, then

$$x^* = \arg\min_{x\in\mathbb{R}^n} f(x) \quad \text{s.t.} \quad c(x) \ge 0$$

$$\Rightarrow \quad \text{KKT:} \quad \exists \lambda \in \mathbb{R}^m \text{ s.t.}$$
$$\begin{cases} \nabla f(x^*) - \nabla c(x^*)\lambda \ = \ 0 \\ 0 \le \lambda \quad \perp \quad c(x^*) \ge 0 \end{cases}.$$

If, in addition, the variables x are nonnegative, then the KKT conditions can be transformed into an NCP, where the first set of complementary conditions is associated with the unknowns x and the second set of complementarity conditions is associated with the Lagrange multipliers λ, namely

$$x^* = \arg\min_{0\le x\in\mathbb{R}^n} f(x) \quad \text{s.t.} \quad c(x) \ge 0$$

$$\Rightarrow \quad \text{NCP:} \quad \exists \lambda \in \mathbb{R}^m \text{ s.t.}$$
$$\begin{cases} 0 \le x^* & \perp \quad \nabla f(x^*) - \nabla c(x^*)\lambda \ge 0 \\ 0 \le \lambda & \perp \quad c(x^*) \ge 0 \end{cases}.$$

Furthermore, suppose that the objective f and the constraints c are convex functions. Then the optimization problem is convex and the KKT (NCP) solution is the unique global optimal solution of the optimization problem (with nonnegative variables). Figure 8 is an example of such a problem. We want to minimize a quadratic function subject to a linear inequality and nonnegative variable. We can introduce the Lagrange multiplier λ associated with the constraint and write down the KKT conditions as a system of two complementarity conditions. In the AMPL code, we select the PATH solver and declare the variables x and λ, followed by the two complementarity conditions.

$$\min_{x\ge 0}\ x^2 - 3x + 2$$
$$\text{s.t. } x - 2 \ge 0$$

$$\Leftrightarrow \quad 0 \le x \ \perp \ 2x - 3 - \lambda \ge 0$$
$$0 \le \lambda \ \perp \ x - 2 \ge 0$$

```
$ ampl CP2.ampl
x = 2
```

```
# File: CP2.ampl
option solver pathampl;
var x := 10;
var lambda := 10;


cond1:
 0 <= x      complements 2 * x - 3 - lambda >= 0;
cond2:
 0 <= lambda complements x >= 2;


solve;
display x;
```

Figure 8: KKT cast as an NCP in 2 variables.

4.2 KKT to MCP

Again, in broad strokes, if we are given bounds $l, u \in \mathbb{R}^n$ for the unknown x, and smooth functions $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$, then a local optimum x^* satisfies the KKT conditions, which implies an MCP:

$$x^* = \arg \min_{l \le x \le u} f(x) \quad \text{s.t.} \quad c(x) \ge 0$$

$$\Rightarrow \quad \text{MCP:} \quad \exists \lambda \in \mathbb{R}^m \text{ s.t.}$$
$$\begin{cases} l \le x \le u & \perp \quad \nabla f(x) - \nabla c(x)\lambda \\ 0 \le \lambda & \perp \quad c(x) \ge 0 \end{cases}.$$

The first set of complementary conditions involves the unknown x in double inequalities, which are orthogonal to the difference between the gradient of f and the product of the Jacobian of c and the Lagrange multipliers λ. The second set of complementarity conditions is associated with λ.

Suppose further that the objective f and the constraint functions c are convex. Then the MCP solution is the unique global optimal solution of the convex optimization problem. The case study CIM-EARTH in the next section demonstrates how this technique can be applied.

5 Case study: CIM-EARTH

CIM-EARTH (Community Integrated Model of Economic and Resource Trajectories for Humankind) [5] offers software libraries for specifying and solving the computable general equilibrium (CGE) models using PATH.

It contains two open-source frameworks ASCEF [2] and OSCEF [16], written in AMPL and C++ respectively. In both frameworks, input data, model output, emulation algorithms, and validation procedures are freely available and easily extensible.

In the rest of this section, we define our canonical CGE model and highlight some parts in the AMPL/PATH scalar models generated by ASCEF and OSCEF.

5.1 CGE model

Our CGE model seeks the Walrasian equilibrium prices and quantities of inputs and outputs that maximize producer profits and consumer utilities and simultaneously satisfy a set of market-clearing conditions.

This example is taken from [4, Sections 2 and 3] with two sectors (material labeled "m" and energy industry "e") and two factors (capital labeled "K" and labor "L"). Here we assume no consumer savings, which would require an additional currency market or CGDS sector to be introduced into the hypothetical economy. The utility tree [4, Figure 2.2] is reduced to a tree with only two children "Materials" and "Energy".

For each output, we maximize the difference between after-tax revenue and after-tax input costs, with the constraints that all the unknowns are nonnegative and that the output (y_m or y_e) is to be bounded by a given (nested) production function of inputs used in the production process.

For consumers as a whole, we want to maximize their utility (or satisfaction) x^c, which is bounded above by a production function of consumer-demanded inputs and savings (zero in this example). The model also restricts consumer expenses to be capped by their incomes and the variables to be nonnegative.

Further, the model requires the nonnegative prices of products and factors to be complementary to their excess supplies.

The resultant, simplified calibrated CGE is as follows:

$$\begin{aligned}
&\text{Material industry:}\\
&\Pi_m = \max_{\mathbf{y}_m \geq 0,\ \mathbf{x}_i^m \geq 0} \bar{r}_m \mathbf{p}_m \mathbf{y}_m - \bar{e}_m^m \mathbf{p}_m \mathbf{x}_m^m - \bar{e}_e^m \mathbf{p}_e \mathbf{x}_e^m - \bar{e}_K^m \mathbf{p}_K \mathbf{x}_K^m - \bar{e}_L^m \mathbf{p}_L \mathbf{x}_L^m\\
&\qquad \text{s.t.} \quad (\mathbf{y}_m)^{\rho^m} \leq \theta_{KL}^m (\mathbf{x}_{KL}^m)^{\rho^m} + \theta_{me}^m (\mathbf{x}_{me}^m)^{\rho^m}\\
&\qquad\qquad (\mathbf{x}_{KL}^m)^{\rho_{KL}^m} \leq \theta_K^m (\mathbf{x}_K^m)^{\rho_{KL}^m} + \theta_L^m (\mathbf{x}_L^m)^{\rho_{KL}^m}\\
&\qquad\qquad (\mathbf{x}_{me}^m)^{\rho_{me}^m} \leq \theta_m^m (\mathbf{x}_m^m)^{\rho_{me}^m} + \theta_e^m (\mathbf{x}_e^m)^{\rho_{me}^m}\\
&\text{Energy industry:}\\
&\Pi_e = \max_{\mathbf{y}_e \geq 0,\ \mathbf{x}_i^e \geq 0} \bar{r}_e \mathbf{p}_e \mathbf{y}_e - \bar{e}_K^e \mathbf{p}_K \mathbf{x}_K^e - \bar{e}_L^e \mathbf{p}_L \mathbf{x}_L^e\\
&\qquad \text{s.t.} \quad (\mathbf{y}_e)^{\rho_{KL}^e} \leq \theta_K^e (\mathbf{x}_K^e)^{\rho_{KL}^e} + \theta_L^e (\mathbf{x}_L^e)^{\rho_{KL}^e}\\
&\text{Consumers utility:}\\
&\max_{0 \leq \mathbf{y}_i \leq \bar{\mathbf{y}}_i,\ \mathbf{x}_i^c \geq 0} \mathbf{x}^c\\
&\qquad \text{s.t.} \quad (\mathbf{x}^c)^{\rho_{me}^c} \leq \theta_m^c (\mathbf{x}_m^c)^{\rho_{me}^c} + \theta_e^c (\mathbf{x}_e^c)^{\rho_{me}^c}\\
&\qquad\qquad \bar{e}_m^c \mathbf{p}_m \mathbf{x}_m^c + \bar{e}_e^c \mathbf{p}_e \mathbf{x}_e^c \leq \bar{r}_L \mathbf{p}_L \mathbf{y}_L + \bar{r}_K \mathbf{p}_K \mathbf{y}_K + \Pi_m + \Pi_e\\
&\text{Market conditions:}\\
&\qquad 0 \leq \mathbf{p}_m \quad \perp \quad \bar{r}_m \mathbf{y}_m \geq \bar{e}_m^m \mathbf{x}_m^m + \bar{e}_m^c \mathbf{x}_m^c\\
&\qquad 0 \leq \mathbf{p}_e \quad \perp \quad \bar{r}_e \mathbf{y}_e \geq \bar{e}_e^m \mathbf{x}_e^m + \bar{e}_e^c \mathbf{x}_e^c\\
&\qquad 0 \leq \mathbf{p}_K \quad \perp \quad \bar{r}_K \mathbf{y}_K \geq \bar{e}_K^m \mathbf{x}_K^m + \bar{e}_K^e \mathbf{x}_K^e\\
&\qquad 0 \leq \mathbf{p}_L \quad \perp \quad \bar{r}_L \mathbf{y}_L \geq \bar{e}_L^m \mathbf{x}_L^m + \bar{e}_L^e \mathbf{x}_L^e
\end{aligned}$$

Converting this CGE problem to an MCP, altogether we have 27 complementarily conditions:

$$\begin{aligned}
&\text{Material industry:}\\
&\qquad \Pi_m \quad \perp \quad \Pi_m = \bar{r}_m \mathbf{p}_m \mathbf{y}_m - \bar{e}_m^m \mathbf{p}_m \mathbf{x}_m^m - \bar{e}_e^m \mathbf{p}_e \mathbf{x}_e^m - \bar{e}_K^m \mathbf{p}_K \mathbf{x}_K^m - \bar{e}_L^m \mathbf{p}_L \mathbf{x}_L^m\\
&\qquad 0 \leq \boldsymbol{\lambda}^m \quad \perp \quad \theta_{KL}^m (\mathbf{x}_{KL}^m)^{\rho^m} + \theta_{me}^m (\mathbf{x}_{me}^m)^{\rho^m} - (\mathbf{y}_m)^{\rho^m} \geq 0
\end{aligned}$$

$$\begin{array}{rcl}
0 \leq \boldsymbol{\lambda}^m_{KL} & \perp & \theta^m_K(\mathbf{x}^m_K)^{\rho^m_{KL}} + \theta^m_L(\mathbf{x}^m_L)^{\rho^m_{KL}} - (\mathbf{x}^m_{KL})^{\rho^m_{KL}} \geq 0 \\
0 \leq \boldsymbol{\lambda}^m_{me} & \perp & \theta^m_m(\mathbf{x}^m_m)^{\rho^m_{me}} + \theta^m_e(\mathbf{x}^m_e)^{\rho^m_{me}} - (\mathbf{x}^m_{me})^{\rho^m_{me}} \geq 0 \\
0 \leq \mathbf{y}_m & \perp & \rho^m(\mathbf{y}_m)^{\rho^m-1}\boldsymbol{\lambda}^m - \bar{r}_m\mathbf{p}_m \geq 0 \\
0 \leq \mathbf{x}^m_m & \perp & \bar{e}^m_m\mathbf{p}_m - \theta^m_m\rho^m_{me}(\mathbf{x}^m_m)^{\rho^m_{me}-1}\boldsymbol{\lambda}^m_{me} \geq 0 \\
0 \leq \mathbf{x}^m_e & \perp & \bar{e}^m_e\mathbf{p}_e - \theta^m_e\rho^m_{me}(\mathbf{x}^m_e)^{\rho^m_{me}-1}\boldsymbol{\lambda}^m_{me} \geq 0 \\
0 \leq \mathbf{x}^m_K & \perp & \bar{e}^m_K\mathbf{p}_K - \theta^m_K\rho^m_{KL}(\mathbf{x}^m_K)^{\rho^m_{KL}-1}\boldsymbol{\lambda}^m_{KL} \geq 0 \\
0 \leq \mathbf{x}^m_L & \perp & \bar{e}^m_L\mathbf{p}_L - \theta^m_L\rho^m_{KL}(\mathbf{x}^m_L)^{\rho^m_{KL}-1}\boldsymbol{\lambda}^m_{KL} \geq 0 \\
0 \leq \mathbf{x}^m_{KL} & \perp & \rho^m_{KL}(x^m_{KL})^{\rho^m_{KL}-1}\boldsymbol{\lambda}^m_{KL} - \theta^m_{KL}\rho^m(\mathbf{x}^m_{KL})^{\rho^m-1}\boldsymbol{\lambda}^m \geq 0 \\
0 \leq \mathbf{x}^m_{me} & \perp & \rho^m_{me}(x^m_{me})^{\rho^m_{me}-1}\boldsymbol{\lambda}^m_{me} - \theta^m_{me}\rho^m(\mathbf{x}^m_{me})^{\rho^m-1}\boldsymbol{\lambda}^m \geq 0 \\
\text{Energy industry:} & & \\
\Pi_e & \perp & \Pi_e = \bar{r}_e\mathbf{p}_e\mathbf{y}_e - \bar{e}^e_K\mathbf{p}_K\mathbf{x}^e_K - \bar{e}^e_L\mathbf{p}_L\mathbf{x}^e_L \\
0 \leq \boldsymbol{\lambda}^e_{KL} & \perp & \theta^e_K(\mathbf{x}^e_K)^{\rho^e_{KL}} + \theta^e_L(\mathbf{x}^e_L)^{\rho^e_{KL}} - (\mathbf{y}_e)^{\rho^e_{KL}} \geq 0 \\
0 \leq \mathbf{y}_e & \perp & \rho^e_{KL}(\mathbf{y}_e)^{\rho^e_{KL}-1}\boldsymbol{\lambda}^e_{KL} - \bar{r}_e\mathbf{p}_e \geq 0 \\
0 \leq \mathbf{x}^e_K & \perp & \bar{e}^e_K\mathbf{p}_K - \theta^e_K\rho^e_{KL}(\mathbf{x}^e_K)^{\rho^e_{KL}-1}\boldsymbol{\lambda}^e_{KL} \geq 0 \\
0 \leq \mathbf{x}^e_L & \perp & \bar{e}^e_L\mathbf{p}_L - \theta^e_L\rho^e_{KL}(\mathbf{x}^e_L)^{\rho^e_{KL}-1}\boldsymbol{\lambda}^e_{KL} \geq 0 \\
\text{Consumers utility:} & & \\
0 \leq \boldsymbol{\lambda}^c_{me} & \perp & \theta^c_m(\mathbf{x}^c_m)^{\rho^c_{me}} + \theta^c_e(\mathbf{x}^c_e)^{\rho^c_{me}} - (\mathbf{x}^c)^{\rho^c_{me}} \geq 0 \\
0 \leq \boldsymbol{\lambda}^c_E & \perp & \bar{r}_L\mathbf{p}_L\mathbf{y}_L + \bar{r}_K\mathbf{p}_K\mathbf{y}_K + \Pi_m + \Pi_e - (\bar{e}^c_m\mathbf{p}_m\mathbf{x}^c_m + \bar{e}^c_e\mathbf{p}_e\mathbf{x}^c_e) \geq 0 \\
0 \leq \mathbf{x}^c & \perp & \rho^c_{me}(\mathbf{x}^c)^{\rho^c_{me}-1}\boldsymbol{\lambda}^c_{me} - 1 \geq 0 \\
0 \leq \mathbf{x}^c_m & \perp & \bar{e}^c_m\mathbf{p}_m\boldsymbol{\lambda}^c_E - \theta^c_m\rho^c_{me}(\mathbf{x}^c_m)^{\rho^c_{me}-1}\boldsymbol{\lambda}^c_{me} \geq 0 \\
0 \leq \mathbf{x}^c_e & \perp & \bar{e}^c_e\mathbf{p}_e\boldsymbol{\lambda}^c_E - \theta^c_e\rho^c_{me}(\mathbf{x}^c_e)^{\rho^c_{me}-1}\boldsymbol{\lambda}^c_{me} \geq 0 \\
0 \leq \mathbf{y}_K \leq \bar{\mathbf{y}}_K & \perp & -\bar{r}_K\mathbf{p}_K\boldsymbol{\lambda}^c_E \\
0 \leq \mathbf{y}_L \leq \bar{\mathbf{y}}_L & \perp & -\bar{r}_L\mathbf{p}_L\boldsymbol{\lambda}^c_E \\
\text{Market conditions:} & & \\
0 \leq \mathbf{p}_m & \perp & \bar{r}_m\mathbf{y}_m \geq \bar{e}^m_m\mathbf{x}^m_m + \bar{e}^c_m\mathbf{x}^c_m \\
0 \leq \mathbf{p}_e & \perp & \bar{r}_e\mathbf{y}_e \geq \bar{e}^m_e\mathbf{x}^m_e + \bar{e}^c_e\mathbf{x}^c_e \\
0 \leq \mathbf{p}_K & \perp & \bar{r}_K\mathbf{y}_K \geq \bar{e}^m_K\mathbf{x}^m_K + \bar{e}^e_K\mathbf{x}^e_K \\
0 \leq \mathbf{p}_L & \perp & \bar{r}_L\mathbf{y}_L \geq \bar{e}^m_L\mathbf{x}^m_L + \bar{e}^e_L\mathbf{x}^e_L.
\end{array}$$

5.2 ASCEF

In ASCEF, the following is a segment of AMPL code for printing out the market-clearing complementarity conditions. Two nested `for` loops and three `printf` statements are shown here. Each time the outer loop is entered, a complementarity condition is printed. The first `printf` statement prints a constraint label. The second `printf` command prints the LHS inequality and the AMPL keyword `complements`. In the inner loop, the third `printf` outputs the RHS inequality.

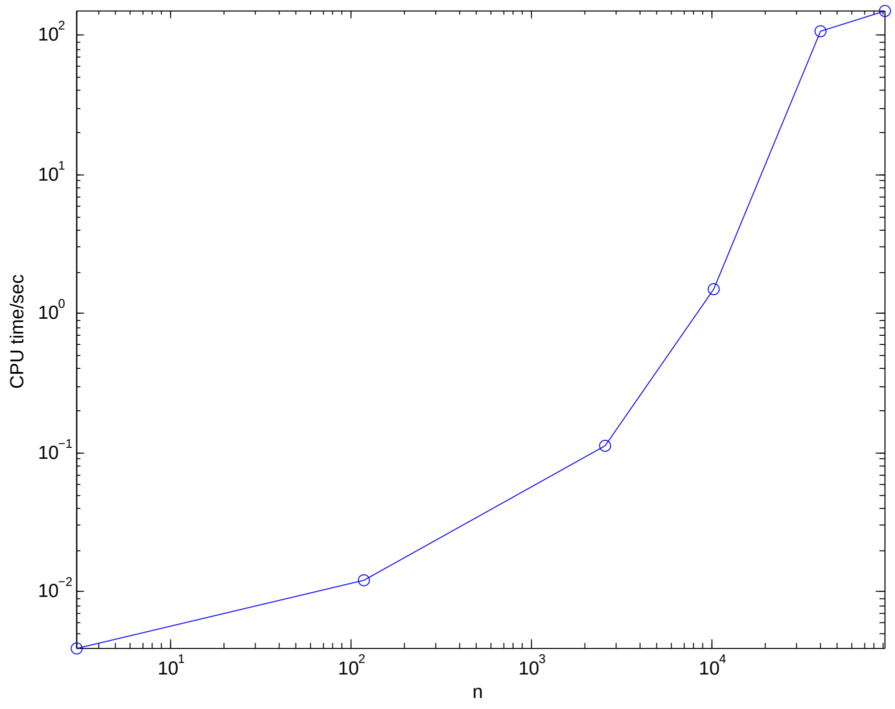


Figure 9: Runtime results from modified AMPL/PATH code on a similar but simpler CGE problem of Cai and Judd (not published).

```
for {k in Region_Commodities[r] diff Homogenous_Commodities} {
  printf "  Market[%s,%s]\n", r, k;                 # label
  printf "  0 <= price[%s,%s] complements", r, k;   # LHS complements
  for {p in Producers, rp in Producers_Regions[p]:
      (r,k) in Producers_Outputs[p,rp]} {           # RHS
    printf " + %5.4e*out[%s,%s,%s,%s]",
        Producers_Revenues[p,rp,r,k]/Markets_Scale[r,k], p, rp, r, k;
  }
}
```

5.3 OSCEF

Appendix A is a partial listing of OSCEF-generated AMPL code for our toy CGE model.

5.4 Numerical results

Figure 9 plots the CPU time dedicated to solving a series of CGE models with increasing number of variables n using PATH/AMPL. It took less than 3 minutes for $90,600$ variables/complementarity constraints, demonstrating the power and efficiency of PATH on this class of problems.

6 AMPL/PATH options and log files

6.1 AMPL options

Many options are available in AMPL. We have seen how to choose PATH as the solver using the option "`solver`":

```
option solver pathampl;
```

In addition, it is highly recommended to turn off the AMPL presolver (in order to use the PATH presolver), by invoking the AMPL option "`presolve`" as follows:

```
option presolve 0;
```

6.2 PATH options

A user can save PATH outputs and options in text files by a single statement:

```
option path_options "logfile = path.log optfile = path.opt";
```

For example, to set or change the convergence tolerance to 10^{-8} in PATH, we can use the PATH option "`convergence_tolerance`" as follows:

```
option path_options "convergence_tolerance = 1e-8";
```

Actually, only the first three letters of each part in an option name matter. Thus we could simply use "`con_tol`" instead:

```
option path_options "con_tol = 1e-8";
```

For reference and completeness, we include in Appendix B the table of key PATH options taken directly from the solver's website [**?**].

7 PATH log file

After AMPL/PATH has attempted to solve a CP, we can check the output log file to see whether the solver has converged to a local solution. Appendix C is an example log file. First we can check whether PATH reports "`Solution found`" at exit. Then we may check that the final residual magnitude is indeed smaller than the prescribed convergence tolerance. Measures of residual norms differ (e.g., `Inf-Norm of Complementarity`, . . . , `Two-Norm of Grad Fischer Fcn`), and they are often close in magnitude if the solver has converged to a solution. If the solver has not converged, a user can examine the `residual` column in the `Iteration Log` to see whether it gives any hints on what PATH options to fine tune.

8 Conclusion

We have presented about ten scalar AMPL programs, starting with toy exmaples of linear equations, nonlinear optimization problems, and complementarity problems. They lead up to two automated frameworks ASCEF and OSCEF in CIM-EARTH that generate large, scalar AMPL models. Our AMPL sample programs are available for download at [16].

A AMPL code

```
option path_options "conv_tol = 1e-8";
option solver pathampl;
option presolve 0;
option log_file 'cimearth.log';
param v0 default 10;

var Profit_m_USA := v0;
var lambda__m_USA := v0;
var lambda__m_me_USA := v0;
var lambda__m_KL_USA := v0;
var y_m_USA := v0;
var x__m_me_USA := v0;
var x__m_m_USA := v0;
var x__m_e_USA := v0;
var x__m_KL_USA := v0;
var x__m_K_USA := v0;
var x__m_L_USA := v0;
var Profit_e_USA := v0;
var lambda__e_USA := v0;
var y_e_USA := v0;
var x__e_K_USA := v0;
var x__e_L_USA := v0;

var lambda__c_USA := v0;
var lambda__c_E_USA := v0;
var x__c_USA := v0;
var x__c_m_USA := v0;
var x__c_e_USA := v0;
var y_L_USA := v0;
var y_K_USA := v0;

var p_m_USA := v0;
var p_e_USA := v0;
var p_L_USA := v0;
var p_K_USA := v0;


SectorProfit_m_USA:
  Profit_m_USA complements Profit_m_USA = (4 - 0) * p_m_USA *
y_m_USA - (1 + 0) * p_m_USA * x__m_m_USA - (1 + 0) * p_e_USA *
x__m_e_USA - (1 + 0) * p_K_USA * x__m_K_USA - (1 + 0) * p_L_USA *
x__m_L_USA;


Sectorlambda__m_USA:
  0 <= lambda__m_USA complements y_m_USA^0.5 <= 0.5 * (1 *
x__m_me_USA)^0.5 + 0.5 * (1 * x__m_KL_USA)^0.5;

Sectorlambda__m_me_USA:
  0 <= lambda__m_me_USA complements x__m_me_USA^0.5 <= 0.5 * (1 *
x__m_m_USA)^0.5 + 0.5 * (1 * x__m_e_USA)^0.5;

Sectorlambda__m_KL_USA:
  0 <= lambda__m_KL_USA complements x__m_KL_USA^0.5 <= 0.5 * (1 *
x__m_K_USA)^0.5 + 0.5 * (1 * x__m_L_USA)^0.5;

Sectory_m_USA:
   0 <= y_m_USA complements 0.5 * (y_m_USA)^-0.5 * lambda__m_USA -
(4 - 0) * p_m_USA >= 0;

Sectorx__m_me_USA:
   0 <= x__m_me_USA complements 0.5 * (x__m_me_USA)^-0.5 *
lambda__m_me_USA - 0.5 * 0.5 * (x__m_me_USA)^-0.5 * lambda__m_USA
>= 0;

Sectorx__m_m_USA:
   0 <= x__m_m_USA complements (1 + 0) * p_m_USA - 0.5 * 0.5 *
(x__m_m_USA)^-0.5 * lambda__m_me_USA >= 0;

Sectorx__m_e_USA:
   0 <= x__m_e_USA complements (1 + 0) * p_e_USA - 0.5 * 0.5 *
(x__m_e_USA)^-0.5 * lambda__m_me_USA >= 0;

Sectorx__m_KL_USA:
   0 <= x__m_KL_USA complements 0.5 * (x__m_KL_USA)^-0.5 *
lambda__m_KL_USA - 0.5 * 0.5 * (x__m_KL_USA)^-0.5 * lambda__m_USA
>= 0;

Sectorx__m_K_USA:
   0 <= x__m_K_USA complements (1 + 0) * p_K_USA - 0.5 * 0.5 *
(x__m_K_USA)^-0.5 * lambda__m_KL_USA >= 0;

Sectorx__m_L_USA:
   0 <= x__m_L_USA complements (1 + 0) * p_L_USA - 0.5 * 0.5 *
(x__m_L_USA)^-0.5 * lambda__m_KL_USA >= 0;

SectorProfit_e_USA:
  Profit_e_USA complements Profit_e_USA = (2 - 0) * p_e_USA *
y_e_USA - (1 + 0) * p_K_USA * x__e_K_USA - (1 + 0) * p_L_USA *
x__e_L_USA;


Sectorlambda__e_USA:
  0 <= lambda__e_USA complements y_e_USA^0.5 <= 0.5 * (1 *
x__e_K_USA)^0.5 + 0.5 * (1 * x__e_L_USA)^0.5;

Sectory_e_USA:
```

B PATH options

Option	Default	Explanation
convergence_tolerance	1e-6	Stopping criterion
crash_iteration_limit	50	Maximum iterations allowed in crash
crash_method	pnewton	pnewton or none
crash_minimum_dimension	1	Minimum problem dimension required to perform crash
crash_nbchange_limit	1	Number of changes to the basis allowed
cumulative_iteration_limit	10000	Maximum minor iterations allowed
lemke_start	automatic	Frequency of lemke starts (automatic, first, always)
major_iteration_limit	500	Maximum major iterations allowed
minor_iteration_limit	1000	Maximum minor iterations allowed in each major iteration
nms	yes	Allow searching, watchdoging, and non-monotone descent
nms_initial_reference_factor	20	Controls size of initial reference value
nms_memory_size	10	Number of reference values kept
nms_mstep_frequency	10	Frequency at which m steps are performed
nms_searchtype	path	path or line
output	yes	Turn output on or off. If output is turned off, selected parts can be turned back on.
output_crash_iterations	yes	Output information on crash iterations
output_crash_iterations_frequency	1	Frequency at which crash iteration log is printed
output_errors	yes	Output error messages
output_linear_model	no	Output linear model each major iteration
output_major_iterations	yes	Output information on major iterations
output_major_iterations_frequency	1	Frequency at which major iteration log is printed
output_minor_iterations	yes	Output information on minor iterations
output_minor_iterations_frequency	500	Frequency at which minor iteration log is printed
output_options	no	Output all options and their values
output_warnings	no	Output warning messages
proximal_perturbation	0.0	Initial perturbation
restart_limit	3	Maximum number of restarts (0 - 3)
time_limit	3600	Maximum number of seconds algorithm is allowed to run

C Sample AMPL/PATH log

```
Path 4.7.03: Path 4.7.03 (Wed Sep  5 16:53:12 2012)
Written by Todd Munson, Steven Dirkse, and Michael Ferris

INITIAL POINT STATISTICS
Maximum of X. . . . . . . . . . .  2.0000e+00 var: (_svar[1])
Maximum of F. . . . . . . . . . .  1.0000e+00 eqn: (_scon[1])
Maximum of Grad F . . . . . . . .  2.0000e+00 eqn: (_scon[1])
                                              var: (_svar[1])

INITIAL JACOBIAN NORM STATISTICS
Maximum Row Norm. . . . . . . . .  2.0000e+00 eqn: (_scon[1])
Minimum Row Norm. . . . . . . . .  2.0000e+00 eqn: (_scon[1])
Maximum Column Norm . . . . . . .  2.0000e+00 var: (_svar[1])
Minimum Column Norm . . . . . . .  2.0000e+00 var: (_svar[1])

Crash Log
major  func  diff  size  residual    step       prox   (label)
    0     0             1.1716e+00             0.0e+00 (_scon[1])
    1     1     1     1 2.7752e-01  1.0e+00    0.0e+00 (_scon[1])
    2     2     0     1 6.3406e-02  1.0e+00    0.0e+00 (_scon[1])
pn_search terminated: no basis change.

Major Iteration Log
major minor  func  grad  residual    step  type prox    inorm  (label)
    0     0     3     3 6.3406e-02           I 0.0e+00 6.3e-02 (_scon[1])
    1     1     4     4 1.5654e-02  1.0e+00 SO 0.0e+00 1.6e-02 (_scon[1])
    2     1     5     5 3.9072e-03  1.0e+00 SO 0.0e+00 3.9e-03 (_scon[1])
    3     1     6     6 9.7659e-04  1.0e+00 SO 0.0e+00 9.8e-04 (_scon[1])
    4     1     7     7 2.4414e-04  1.0e+00 SO 0.0e+00 2.4e-04 (_scon[1])
    5     1     8     8 6.1035e-05  1.0e+00 SO 0.0e+00 6.1e-05 (_scon[1])
    6     1     9     9 1.5259e-05  1.0e+00 SO 0.0e+00 1.5e-05 (_scon[1])
    7     1    10    10 3.8147e-06  1.0e+00 SO 0.0e+00 3.8e-06 (_scon[1])
    8     1    11    11 9.5367e-07  1.0e+00 SO 0.0e+00 9.5e-07 (_scon[1])

FINAL STATISTICS
Inf-Norm of Complementarity . .  9.5461e-07 eqn: (_scon[1])
Inf-Norm of Normal Map. . . . .  9.5367e-07 eqn: (_scon[1])
Inf-Norm of Minimum Map . . . .  9.5367e-07 eqn: (_scon[1])
Inf-Norm of Fischer Function. .  9.5367e-07 eqn: (_scon[1])
Inf-Norm of Grad Fischer Fcn. .  1.8626e-09 eqn: (_scon[1])
Two-Norm of Grad Fischer Fcn. .  1.8626e-09

FINAL POINT STATISTICS
Maximum of X. . . . . . . . . . .  1.0010e+00 var: (_svar[1])
Maximum of F. . . . . . . . . . .  9.5367e-07 eqn: (_scon[1])
Maximum of Grad F . . . . . . . .  1.9531e-03 eqn: (_scon[1])
                                              var: (_svar[1])

 ** EXIT - solution found.

Major Iterations. . . . 8
Minor Iterations. . . . 8
Restarts. . . . . . . . 0
Crash Iterations. . . . 2
Gradient Steps. . . . . 0
Function Evaluations. . 11
Gradient Evaluations. . 11
Basis Time. . . . . . . 0.000000
Total Time. . . . . . . 0.000000
Residual. . . . . . . . 9.536743e-07
Solution found.
10 iterations (2 for crash); 8 pivots.
11 function, 11 gradient evaluations.
x = 1.00098
```

References

[1] AMPL website. http://http://www.ampl.com/, 2013.

[2] ASCEF. http://www.rdcep.org/ascef, 2013.

[3] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, 2013.

[4] Joshua Elliott, Ian Foster, Kenneth Judd, Elisabeth Moyer, and Todd Munson. CIM-EARTH: Community integrated model of economic and resource trajectories for humankind. Tech. Rep. ANL/MCS-TM-307, Argonne National Laboratory, Argonne, Illinois, 2010.

[5] Joshua Elliott, Ian Foster, Kenneth Judd, Elisabeth Moyer, and Todd Munson. CIM-EARTH: Framework and case study. *BEJEAP*, 10(2 (Symposium)), 2010.

[6] Michael Ferris, Robert Fourer, and David Gay. Expressing complementarity problems in an algebraic modeling language and communicating them to solvers. *SIAM J. Optim.*, 9(4):991–1009, 1999. Dedicated to John E. Dennis, Jr., on his 60th birthday.

[7] Michael Ferris and Todd Munson. Interfaces to PATH 3.0: Design, implementation and usage. *Computational Optimization and Applications*, 12(1):207–227, 1999.

[8] Robert Fourer, David M Gay, and Brian Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole–Thomson Learning, Pacific Grove, California, 2nd edition, 2003.

[9] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. Maintaining LU factors of a general sparse matrix. *Linear Algebra and its Applications*, 88:239–270, 1987.

[10] KNITRO website. http://www.ziena.com/knitro.htm, 2013.

[11] LUSOL: Sparse LU for $Ax = b$. http://www.stanford.edu/group/SOL/software/lusol.html, 2013.

[12] MINOS website. http://www.sbsi-sol-optimize.com/asp/sol_product_minos.htm, 2013.

[13] NEOS AMPL/PATH website. http://www.neos-server.org/neos/solvers/cp:PATH/AMPL.html, 2013.

[14] NEOS website. http://www.neos-server.org/neos/, 2013.

[15] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2nd edition, 2006.

[16] OSCEF website. http://www.rdcep.org/research/open-source-cim-earth-framework-oscef, 2013.

Acknowledgments

We acknowledge that the work was partially completed while the authors were visiting the Institute for Mathematical Sciences (IMS), National University of Singapore in 2012. We also acknowledge SIAM's financial support to travel to IMS. In addition, we thank Alison Brizus, Yongyang Cai, Michaela Carey, Richard Cottle, Michael Ferris, Ian Foster, Ilse Ipsen, Kenneth Judd, Todd Munson, Jorge Nocedal, Jong-Shi Pang, Gail Pieper, Daniel Ralph, Defeng Sun, Kim-Chuan Toh, and Weichung Wang for their interest in and discussion during the development of this work.