Complementarity Problems with AMPL

Todd Munson

Mathematics and Computer Science Division
Argonne National Laboratory

Outline

- Nash Games and Named Problems
- Complementarity Formulation
- Numerical Methods
- Intermediate AMPL
- Conclusion

Nash Games

- **Non-cooperative game played by n individuals**
- **Each selects strategy to optimize their objective**
- **Given strategies selected by other players**
- **Equilibrium when no improvement possible**
- **Characterization of two player equilibrium (x_1^*, x_2^*):**

$$x_1^* \in \begin{cases} \arg\min\limits_{x_1 \geq 0} & f_1(x_1, x_2^*) \\ \text{subject to} & c_1(x_1) \leq 0 \end{cases}$$

$$x_2^* \in \begin{cases} \arg\min\limits_{x_2 \geq 0} & f_2(x_1^*, x_2) \\ \text{subject to} & c_2(x_2) \leq 0 \end{cases}$$

Economic Applications

- **Bi-matrix games**
- **Cournot duopoly models**
- **General equilibrium models**
- **Walrasian equilibrium**
- **Arrow-Debreau models**

Block Gauss-Seidel Method

1. For each player p
   (a) Solve optimization problem

$$\hat{x}_p \in \begin{cases} \arg\min\limits_{x_p \geq 0} & f_p(\bar{x}_p) \\ \text{subject to} & c_p(x_p) \leq 0 \end{cases}$$

   with $\bar{x}_p = (x_1^*, \ldots, x_{p-1}^*, x_p, x_{p+1}^*, \ldots, x_n^*)$

   (b) Update $x_p^* = \hat{x}_p$
2. Check convergence and repeat

- Cannot be applied to every problem
- Method may not converge
- Linear convergence rate

Oligopoly Model

- Each firm chooses output to maximize profit
  - α and η are parameters
  - c_f is unit cost for each firm
  - u is the utility function

$$u = \left(1 + \sum_{f \in F} x_f^{\alpha}\right)^{\frac{\eta}{\alpha}}$$

  - x_f is output for each firm
- In particular, for each $f \in F$

$$x_f^* \in \arg\max_{x_f \geq 0} \left(\frac{\partial u}{\partial x_f} - c_f\right) x_f$$

Model Definition: oligopoly.mod

```
set FIRMS;                                          # Firms in problem

param c {FIRMS};                                    # Unit cost
param alpha > 0;                                    # Constants
param eta > 0;

var x {FIRMS} >= 0 default 0.1;                     # Output
var u =                                             # Utility
  (1 + sum {f in FIRMS} x[f]^alpha)^(eta/alpha);
var du {f in FIRMS} =                               # Price
  eta * (1 + sum {g in FIRMS} x[g]^alpha)^(eta/alpha - 1) * x[f]^(alpha - 1);

maximize profit {f in FIRMS}:
  (du[f] - c[f]) * x[f];

problem opt {f in FIRMS}: profit[f], x[f], du[f];
```

Data Definition: `oligopoly.dat`

```
param: FIRMS :  c :=
         1     0.07
         2     0.08
         3     0.09;

param alpha := 0.999;
param eta := 0.2;
```

Execution Commands: `oligopoly.cmd`

```
# Load model and data
model oligopoly.mod;
data oligopoly.dat;

# Specify solver and options
option solver "minos";
option minos_options "outlev=1";

# Simple Block Gauss-Seidel method
for {iter in 1..10} {
  for {f in FIRMS} {
    problem opt[f];
    solve;
  }
}

printf {f in FIRMS} "Output for firm %2d: % 5.4e\n", f, x[f] > oligopoly.out;
```

Output File: `oligopoly.out`

```
Output for firm  1:  8.3734e-01
Output for firm  2:  5.0719e-01
Output for firm  3:  1.7923e-01
```

Wardropian Equilibrium with Congestion

- Route commodities through network
  - $\mathcal{N}$ denotes the nodes and $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$ the arcs
  - $\mathcal{K}$ denotes the commodities
  - α, β are the congestion parameters
  - b denotes the supply and demand
- Minimize time for each commodity k

$$\min_{x_{*,*,k} \geq 0} \quad \sum_{(i,j) \in \mathcal{A}} \alpha_{i,j} \left(\sum_{k \in \mathcal{K}} x_{i,j,k} \right) + \beta_{i,j} \left(\sum_{k \in \mathcal{K}} x_{i,j,k} \right)^4$$

$$\text{subject to} \quad \sum_{(i,j) \in \mathcal{A}} x_{i,j,k} \leq \sum_{(j,i) \in \mathcal{A}} x_{j,i,k} + b_{i,k} \qquad \forall i \in \mathcal{N}$$

Model Definition: wardrop.mod

```
set NODES;                                          # Nodes in network
set ARCS within NODES cross NODES;                  # Arcs in network
set COMMODITIES := 1..3;                            # Commodities

param b {NODES, COMMODITIES} default 0;             # Supply/demand
param alpha {ARCS} >= 0;                            # Linear part
param beta {ARCS} >= 0;                             # Nonlinear part

var x {ARCS, COMMODITIES} >= 0;                     # Flow on arcs
var f {(i,j) in ARCS} =                             # Total flow
  sum {k in COMMODITIES} x[i,j,k];

minimize time {k in COMMODITIES}:
  sum {(i,j) in ARCS} (alpha[i,j]*f[i,j] + beta[i,j]*f[i,j]^4);

subject to
  conserve {i in NODES, k in COMMODITIES}:
    sum {(i,j) in ARCS} x[i,j,k] <= sum{(j,i) in ARCS} x[j,i,k] + b[i,k];

problem player {k in COMMODITIES}: time[k], {i in NODES} conserve[i,k],
                                   {(i,j) in ARCS} x[i,j,k], f;
```

Data Definition: wardrop.dat

```
set NODES := 1 2 3 4 5;

param: ARCS : alpha beta =
        1 2     1    0.5
        1 3     1    0.4
        2 3     2    0.7
        2 4     3    0.1
        3 2     1    0.0
        3 4     4    0.5
        4 1     5    0.0
        4 5     2    0.1
        5 2     0    1.0;

let b[1,1] :=  7;     # Node 1, Commodity 1 supply
let b[4,1] := -7;     # Node 4, Commodity 1 demand
let b[2,2] :=  3;     # Node 2, Commodity 2 supply
let b[5,2] := -3;     # Node 5, Commodity 2 demand
let b[3,3] :=  5;     # Node 1, Commodity 3 supply
let b[1,3] := -5;     # Node 4, Commodity 3 demand
```

Execution Commands: wardrop.cmd

```
# Load model and data
model wardrop.mod;
data wardrop.dat;

# Specify solver and options
option solver "minos";
option minos_options "outlev=1";

# Simple Block Gauss-Seidel method
for {iter in 1..90} {
  for {k in COMMODITIES} {
    problem player[k];
    solve;
  }
}

for {k in COMMODITIES} {
  printf "Commodity: %d\n", k > wardrop.out;
  printf {(i,j) in ARCS: x[i,j,k] > 0} "%d.%d = % 5.4e\n", i, j, x[i,j,k] > wardrop.out;
  printf "\n" > wardrop.out;
}
```

Output File: wardrop.out

```
Commodity: 1
1.2 =  3.3775e+00
1.3 =  3.6225e+00
2.4 =  4.4169e+00
3.2 =  1.0394e+00
3.4 =  2.5831e+00

Commodity: 2
2.4 =  3.0000e+00
4.5 =  3.0000e+00

Commodity: 3
2.4 =  2.0480e+00
3.2 =  2.0480e+00
3.4 =  2.9520e+00
4.1 =  5.0000e+00
```

Bimatrix Game

- **Player 1**

$$\begin{aligned} \min_{p \geq 0} \quad & p^T \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} q \\ \text{subject to} \quad & e^T p = 1 \end{aligned}$$

- **Player 2**

$$\begin{aligned} \min_{q \geq 0} \quad & p^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} q \\ \text{subject to} \quad & e^T q = 1 \end{aligned}$$

- **Gauss-Seidel cycles**

$$p = (0,1) \Rightarrow q = (1,0) \Rightarrow p = (1,0) \Rightarrow q = (0,1) \Rightarrow p = (0,1)$$

Optimization Theory Review

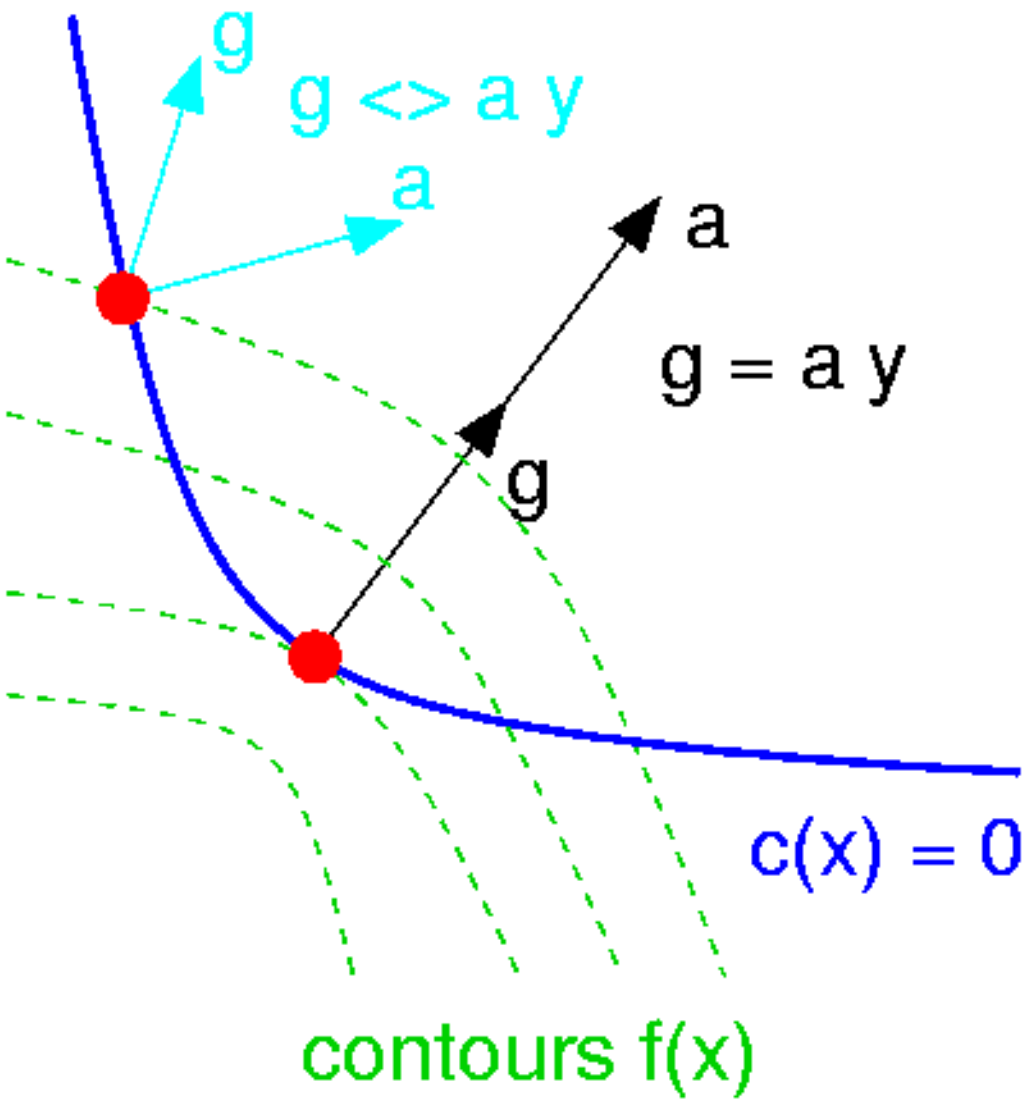
- Assume convex optimization problem
  - $f(\cdot)$ and $c(\cdot)$ are convex function
  - Constraint qualification satisfied
- Then first-order conditions are necessary and sufficient

$$\begin{array}{ll} \min\limits_{x \geq 0} & f(x) \\ \text{subject to} & c(x) \leq 0 \end{array} \quad \Leftrightarrow \quad \begin{array}{lcl} 0 \leq x & \perp & \nabla f(x) + \lambda^T \nabla c(x) \geq 0 \\ 0 \leq \lambda & \perp & -c(x) \geq 0 \end{array}$$

- Notation: $0 \leq a \perp b \geq 0 \Leftrightarrow a \geq 0, b \geq 0, ab = 0$
- Nonlinear complementarity problem

$$0 \leq x \perp F(x) \geq 0$$

Illustration



Application to Nash Games

- Assume each player solves a convex problem
  - $f_1(\cdot, x_2)$ is convex for all x_2
  - $f_2(x_1, \cdot)$ is convex for all x_1
  - $c_1(\cdot)$ and $c_2(\cdot)$ are convex functions
  - Constraint qualification satisfied
- Stacked first-order conditions characterize equilibrium

$$
\begin{aligned}
0 \le x_1 \quad &\perp \quad \nabla_{x_1} f_1(x_1, x_2) + \lambda_1^T \nabla c_1(x_1) \ge 0 \\
0 \le \lambda_1 \quad &\perp \quad -c_1(x_1) \ge 0 \\
0 \le x_2 \quad &\perp \quad \nabla_{x_2} f_2(x_1, x_2) + \lambda_2^T \nabla c_2(x_2) \ge 0 \\
0 \le \lambda_2 \quad &\perp \quad -c_2(x_2) \ge 0
\end{aligned}
$$

- Square nonlinear complementarity problem

Numerical Methods I

- **Josephy-Newton Method for $0 \leq x \perp F(x) \geq 0$**

$$0 \leq x^{k+1} \perp \nabla F(x^k)(x^{k+1} - x^k) + F(x^k) \geq 0$$

  - **PATH (AMPL and GAMS)**
    - **Semismooth merit function**
    - **Nonmonotone line search**
    - **Modified Lemke method**
    - **Crash technique**
  - **MILES (GAMS only)**

Numerical Methods II

- **Semismooth Methods**
  - **NCP Functions $\phi : \Re^2 \rightarrow \Re$**

$$\phi(a, b) = 0 \;\Leftrightarrow\; 0 \leq a \;\perp\; b \geq 0$$

    - **Fischer-Burmeister function**

$$\phi(a, b) = a + b - \sqrt{a^2 + b^2}$$

    - **Reformulate complementarity problem**

$$\Phi(x) = [\phi(x_i, F_i(x))]$$

  - **Apply Newton method to $\Phi(x) = 0$**

$$x^{k+1} = H_k^{-1}(x^k - \Phi(x^k))$$

$$H \in \partial\Phi(x)$$

Numerical Methods III

- Reformulate as optimization problem
  - Minimize complementarity error

$$\begin{aligned} \min_{x \geq 0, s \geq 0} \quad & x^T s \\ \text{subject to} \quad & s = F(x) \end{aligned}$$

  - Many possible formulations

$$\begin{aligned} \min_{x \geq 0, s \geq 0} \quad & 0 \\ \text{subject to} \quad & s = F(x) \\ & x^T s \leq 0 \end{aligned}$$

- Interior-point methods for special cases

Oligopoly Model Revisited

- Each firm chooses output to maximize profit
- u is the utility function

$$u = \left(1 + \sum_{f \in F} x_f^{\alpha}\right)^{\frac{\eta}{\alpha}}$$

- For each $f \in F$

$$x_f^* \in \arg\max_{x_f \geq 0} \left(\frac{\partial u}{\partial x_f} - c_f\right) x_f$$

- Complementarity problem

$$0 \leq x_f \perp c_f - \frac{\partial u}{\partial x_f} - \frac{\partial^2 u}{\partial x_f^2} x_f \geq 0$$

Model Definition: oligcomp.mod

```
set FIRMS;                                          # Firms in problem

param c {FIRMS};                                    # Unit cost
param alpha > 0;                                    # Constants
param eta > 0;

var x {FIRMS} default 0.1;                          # Output (no bounds!)

var s = 1 + sum {f in FIRMS} x[f]^alpha;            # Summation term
var u = s^(eta/alpha);                              # Utility
var du {f in FIRMS} =                               # Price
  eta * s^(eta/alpha - 1) * x[f]^(alpha - 1);
var dudu {f in FIRMS} =                             # Derivative
  eta * (eta - alpha) * s^(eta/alpha - 2) * x[f]^(2 * alpha - 2) +
  eta * (alpha - 1  ) * s^(eta/alpha - 1) * x[f]^(    alpha - 2);

subject to
compl {f in FIRMS}:
  0 <= x[f] complements c[f] - du[f] - dudu[f]*x[f] >= 0;
```

Execution Commands: oligcomp.cmd

```
# Load model and data
model oligcomp.mod;
data oligcomp.dat;

# Specify solver and options
option presolve 0;
option solver "kestrel";
option kestrel_options "solver=path";

# Solve complementarity problem
solve;

printf {f in FIRMS} "Output for firm %2d: % 5.4e\n", f, x[f] > oligcomp.out;
```

Output File: `oligcomp.out`

```
Output for firm  1:  8.3735e-01
Output for firm  2:  5.0720e-01
Output for firm  3:  1.7921e-01
```

Equilibrium for Endowment Economy I

- **Economy with n agents and m commodities**
  - **$e \in \Re^{n \times m}$ are the endowments**
  - **$\alpha, \beta \in \Re^{n \times m}$ are the utility parameters**
  - **$x \in \Re^{n \times m}$ is the amount consumed**
  - **$p \in \Re^{m}$ is the commodity price**

Equilibrium for Endowment Economy II

- Agent i maximizes utility with budget constraint

$$\begin{aligned} \max_{x_{i,*} \geq 0} \quad & \sum_{k=1}^{n} \frac{\alpha_{i,k}(1 + x_{i,k})^{1-\beta_{i,k}}}{1 - \beta_{i,k}} \\ \text{subject to} \quad & \sum_{k=1}^{m} p_k \left(x_{i,k} - e_{i,k}\right) \leq 0 \end{aligned}$$

- Market k sets price for commodity

$$0 \leq p_k \quad \perp \quad \sum_{i=1}^{n} \left(e_{i,k} - x_{i,k}\right) \geq 0$$

- Square complementarity problem

Model Definition I: cge.mod

```
set AGENTS;                                          # Agents
set COMMODITIES;                                     # Commodities

param e {AGENTS, COMMODITIES} >= 0, default 1; # Endowment

param alpha {AGENTS, COMMODITIES} > 0;               # Utilty parameters
param beta {AGENTS, COMMODITIES} > 0;

var x {AGENTS, COMMODITIES};                         # Consumption (no bounds!)
var l {AGENTS} default 1;                            # Multipliers (no bounds!)
var p {COMMODITIES} default 1;                       # Prices (no bounds!)
```

Model Definition II: cge.mod

```
var du {i in AGENTS, k in COMMODITIES} =       # Marginal prices
  alpha[i,k] / (1 + x[i,k])^beta[i,k];

subject to
  optimality {i in AGENTS, k in COMMODITIES}:
    0 <= x[i,k] complements -du[i,k] + p[k] * l[i] >= 0;

  budget {i in AGENTS}:
    0 <= l[i]   complements sum {k in COMMODITIES} p[k]*(e[i,k] - x[i,k]) >= 0;

  market {k in COMMODITIES}:
    0 <= p[k]   complements sum {i in AGENTS} (e[i,k] - x[i,k]) >= 0;
```

Data Definition: cge.dat

```
set AGENTS := Jorge, Sven, Todd;
set COMMODITIES := Books, Cars, Food, Pens;

param alpha : Books  Cars  Food  Pens :=
     Jorge      1     1     1     1
     Sven       1     2     3     4
     Todd       2     1     1     5;

param beta (tr): Jorge  Sven  Todd :=
     Books        1.5    2     0.6
     Cars         1.6    3     0.7
     Food         1.7    2     2.0
     Pens         1.8    2     2.5;
```

Execution Commands: cge.cmd

```
# Load model and data
model cge.mod;
data cge.dat;

# Specify solver and options
option presolve 0;
option solver "kestrel";
option kestrel_options "solver=path";

# Solve the instance
solve;

# Output results
printf {i in AGENTS, k in COMMODITIES} "%5s %5s: % 5.4e\n", i, k, x[i,k] > cge.out;
printf "\n" > cge.out;
printf {k in COMMODITIES} "%5s: % 5.4e\n", k, p[k] > cge.out;
```

Output File: cge.out

```
Jorge Books:  8.9825e-01
Jorge  Cars:  1.4651e+00
Jorge  Food:  1.2021e+00
Jorge  Pens:  6.8392e-01
 Sven Books:  2.5392e-01
 Sven  Cars:  7.2054e-01
 Sven  Food:  1.6271e+00
 Sven  Pens:  1.4787e+00
 Todd Books:  1.8478e+00
 Todd  Cars:  8.1431e-01
 Todd  Food:  1.7081e-01
 Todd  Pens:  8.3738e-01

Books:  1.0825e+01
 Cars:  6.6835e+00
 Food:  7.3983e+00
 Pens:  1.1081e+01
```

Execution Commands: cgenum.cmd

```
# Load model and data
model cge.mod;
data cge.dat;

# Specify solver and options
option presolve 0;
option solver "kestrel";
option kestrel_options "solver=path";

# Solve the instance
drop market['Books'];
fix p['Books'] := 1;
solve;

# Output results
printf {i in AGENTS, k in COMMODITIES} "%5s %5s: % 5.4e\n", i, k, x[i,k] > cgenum.out;
printf "\n" > cgenum.out;
printf {k in COMMODITIES} "%5s: % 5.4e\n", k, p[k] > cgenum.out;
```

Output File: cgenum.out

```
Jorge Books:  8.9825e-01
Jorge  Cars:  1.4651e+00
Jorge  Food:  1.2021e+00
Jorge  Pens:  6.8392e-01
 Sven Books:  2.5392e-01
 Sven  Cars:  7.2054e-01
 Sven  Food:  1.6271e+00
 Sven  Pens:  1.4787e+00
 Todd Books:  1.8478e+00
 Todd  Cars:  8.1431e-01
 Todd  Food:  1.7081e-01
 Todd  Pens:  8.3738e-01

Books:  1.0000e+00
 Cars:  6.1742e-01
 Food:  6.8345e-01
 Pens:  1.0237e+00
```

Extensions

- **Mixed complementarity problems**

$$\ell \leq x \leq u \quad \perp \quad F(x)$$

- **Nonlinear systems of equations**

$$x \text{ free} \quad \perp \quad F(x) = 0$$

- **Generalized relationships**

$$0 \leq G(x) \quad \perp \quad F(x) \geq 0$$

  - **Slacks and constraints added**
  - **Reformulated as mixed complementarity problem**

Pitfalls

- **Nonsquare systems**
  - **Side variables**
  - **Side constraints**
- **Nonideal orientation for equations**
  - **Skew symmetry preferred**
  - **Proximal point perturbation**
- **AMPL presolve**
  - `option presolve 0;`

Conclusion

- Algorithms can be written in AMPL
- Complementarity problems are useful
  - Express equations and inequalities
  - Determine activities
- Some algorithms for solving them
  - Commercially available
  - Usable through NEOS