

Solving for equilibria using a contraction mapping	
• In many instances, we may want to solve for an equilibrium	

Solving for equilibria using a contraction mapping
 In many instances, we may want to solve for an equilibrium

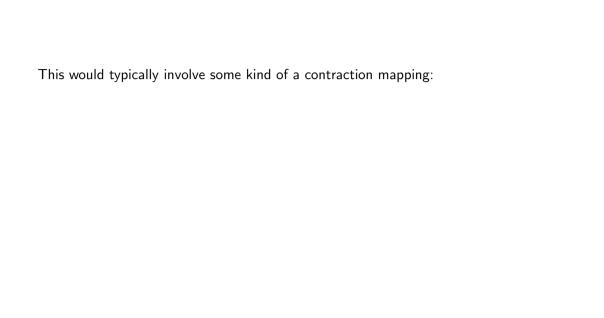
• This is especially common in IO applications, where firms interact strategically

Solving	for	equilibria	using	а	contraction	mapping

• In many instances, we may want to solve for an equilibrium

• This is especially common in IO applications, where firms interact strategically

• The goal is to estimate preference parameters consistent with the equilibrium



This would	typically	involve	some	kind	of a	contraction	mapping:	

• Take a guess at the parameter values

Т	his	would	typica	llv	involve	some	kind	of	а	contraction	mapping:
•		would	c, p.ca	,		501110		\sim .	•	COLLCIACTION	

Take a guess at the parameter values

• Conditional on the parameter values, solve for the equilibrium

This would typically involve some kind of a contraction mapping:

- Take a guess at the parameter values
- Conditional on the parameter values, solve for the equilibrium
- \bullet Update the parameter values, re-solve for the equilibrium, \dots

This would typically involve some kind of a contraction mapping:

- Take a guess at the parameter values
- Conditional on the parameter values, solve for the equilibrium
- Update the parameter values, re-solve for the equilibrium, ...
- Just like in Rust's (1987) NFXP algorithm

\sim 1 ·	_			MADEC
Solving	tor	equilibria	iising	MPEC
20.11.15		cquiiibiia	455	

• An alternative approach to solving for equilibria is MPEC

• An alternative approach to solving for equilibria is MPEC

• MPEC: Mathematical Programming with Equilibrium Constraints

- An alternative approach to solving for equilibria is MPEC
- MPEC: Mathematical Programming with Equilibrium Constraints
- With MPEC, we re-cast the equilibrium as a set of optimization constraints

- An alternative approach to solving for equilibria is MPEC
- MPEC: Mathematical Programming with Equilibrium Constraints
- With MPEC, we re-cast the equilibrium as a set of optimization constraints
- This reduces the need to solve for a fixed point

- An alternative approach to solving for equilibria is MPEC
- MPEC: Mathematical Programming with Equilibrium Constraints
- With MPEC, we re-cast the equilibrium as a set of optimization constraints
- This reduces the need to solve for a fixed point
- Typically the estimation converges much faster

- An alternative approach to solving for equilibria is MPEC
- MPEC: Mathematical Programming with Equilibrium Constraints
- With MPEC, we re-cast the equilibrium as a set of optimization constraints
- This reduces the need to solve for a fixed point
- Typically the estimation converges much faster
 - Because the optimizer sees the constraints and makes "smarter" guesses

ECONOMETRICA

JOURNAL OF THE ECONOMETRIC SOCIETY

An International Society for the Advancement of Economic Theory in its Relation to Statistics and Mathematics

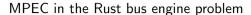
http://www.econometricsociety.org/

Econometrica, Vol. 80, No. 5 (September, 2012), 2213-2230

CONSTRAINED OPTIMIZATION APPROACHES TO ESTIMATION OF STRUCTURAL MODELS

CHE-LIN SU University of Chicago, Booth School of Business, Chicago, IL 60637, U.S.A.

> KENNETH L. JUDD Hoover Institution, Stanford, CA 94305, U.S.A. and NBER



• Compare MPEC with NFXP in the Rust (1987) model

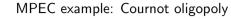
• Compare MPEC with NFXP in the Rust (1987) model

• Su & Judd (2012) initially reported substantial speed improvements

- Compare MPEC with NFXP in the Rust (1987) model
- Su & Judd (2012) initially reported substantial speed improvements
- However: Iskhakov et al. (2016) showed similar performance when both methods are properly optimized

- Compare MPEC with NFXP in the Rust (1987) model
- Su & Judd (2012) initially reported substantial speed improvements
- However: Iskhakov et al. (2016) showed similar performance when both methods are properly optimized
- Key advantages of MPEC: ease of implementation and robustness

- Compare MPEC with NFXP in the Rust (1987) model
- Su & Judd (2012) initially reported substantial speed improvements
- However: Iskhakov et al. (2016) showed similar performance when both methods are properly optimized
- Key advantages of MPEC: ease of implementation and robustness
- The optimizer sees constraints and makes "smarter" guesses



• JuMP, with an add-on package Complementarity.jl, supports MPEC

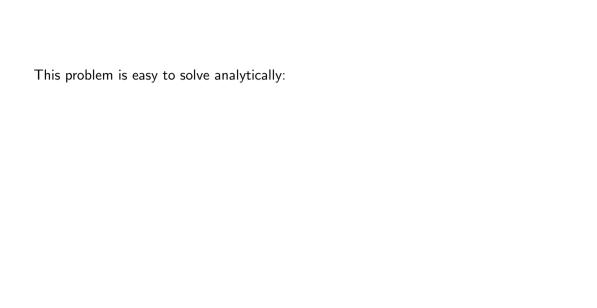
• JuMP, with an add-on package Complementarity.jl, supports MPEC

• Example: *N*-firm symmetric Cournot monopolistic competition

- JuMP, with an add-on package Complementarity.jl, supports MPEC
- Example: *N*-firm symmetric Cournot monopolistic competition
- ullet Each firm i chooses output q_i to maximize profit, subject to q_{-i}

- JuMP, with an add-on package Complementarity.jl, supports MPEC
- Example: N-firm symmetric Cournot monopolistic competition
- Each firm i chooses output q_i to maximize profit, subject to q_{-i}
- ullet Marginal cost c_i is assumed to be constant and equal across all firms

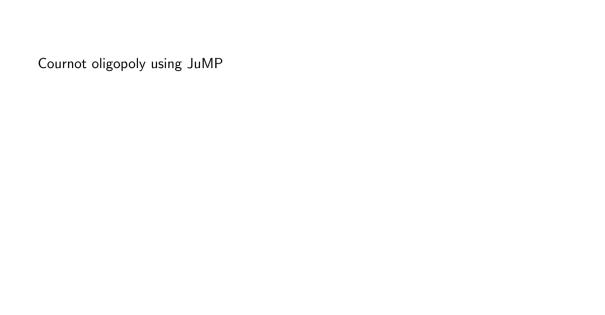
- JuMP, with an add-on package Complementarity.jl, supports MPEC
- Example: N-firm symmetric Cournot monopolistic competition
- Each firm i chooses output q_i to maximize profit, subject to q_{-i}
- \bullet Marginal cost c_i is assumed to be constant and equal across all firms
- Market demand is given by P(Q) = a bQ

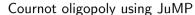


This problem is easy to solve analytically:

 $q^* = \frac{a-c}{b(N+1)}$

 $P^* = \frac{a + Nc}{N + 1}$





• JuMP allows us to formulate this as an MPEC problem

Cournot oligopoly using JuMP

• JuMP allows us to formulate this as an MPEC problem

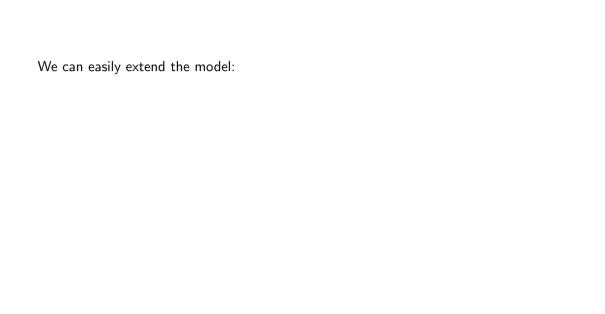
• Firm 1 minimizes negative profit subject to other firms' FOCs

Cournot oligopoly using JuMP

- JuMP allows us to formulate this as an MPEC problem
- Firm 1 minimizes negative profit subject to other firms' FOCs
- We impose symmetry: all firms choose the same output in equilibrium

Cournot oligopoly using JuMP

- JuMP allows us to formulate this as an MPEC problem
- Firm 1 minimizes negative profit subject to other firms' FOCs
- We impose symmetry: all firms choose the same output in equilibrium
- The optimizer finds the equilibrium without nested fixed-point iteration



• Add capacity constraints: $0 \le q_i \le L$

- Add capacity constraints: $0 \le q_i \le L$
- Vary the number of firms N

- Add capacity constraints: $0 \le q_i \le L$
- Vary the number of firms N
- Allow asymmetric costs c_i

- Add capacity constraints: $0 \le q_i \le L$
- Vary the number of firms N
- Allow asymmetric costs c_i
 - Use non-linear demand functions