Why do we need constrained optimization?	
In nonlinear optimization, constraints can be very helpful, for a number of reasons:	

- Numerical stability
  - e.g. optimization will crash if it tries a negative value for a variance parameter

- Numerical stability
  - e.g. optimization will crash if it tries a negative value for a variance parameter
- Make results consistent with economic theory
  - ullet e.g. discount factor  $eta \in [0,1]$  in DDC models, otherwise model is undefined

- Numerical stability
  - e.g. optimization will crash if it tries a negative value for a variance parameter
- Make results consistent with economic theory
  - ullet e.g. discount factor  $eta \in [0,1]$  in DDC models, otherwise model is undefined
- Simplify the problem
  - e.g.  $\beta = 0$  reduces to a static model

- Numerical stability
  - e.g. optimization will crash if it tries a negative value for a variance parameter
- Make results consistent with economic theory
  - ullet e.g. discount factor  $eta \in [0,1]$  in DDC models, otherwise model is undefined
- Simplify the problem
  - e.g.  $\beta = 0$  reduces to a static model
- More quickly solve equilibrium models through a method called MPEC

$$\max_{x} f(x)$$
  
subject to  $g(x) \leq 0$ 

$$\mathcal{L}(x,\lambda) = f(x) - \lambda g(x)$$

$$\max_{x} f(x)$$
  
subject to  
$$g(x) \le 0$$

$$\mathcal{L}(x,\lambda) = f(x) - \lambda g(x)$$

• In the case of optimization, f(x) is our likelihood function; x's are the parameters

$$\max_{x} f(x)$$
  
subject to  
$$g(x) \le 0$$

$$\mathcal{L}(x,\lambda) = f(x) - \lambda g(x)$$

- In the case of optimization, f(x) is our likelihood function; x's are the parameters
- The first-order conditions (FOCs) tell us what the optimal x's are

$$\max_{x} f(x)$$
  
subject to  
$$g(x) \le 0$$

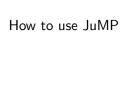
$$\mathcal{L}(x,\lambda) = f(x) - \lambda g(x)$$

- In the case of optimization, f(x) is our likelihood function; x's are the parameters
- The first-order conditions (FOCs) tell us what the optimal x's are
- Also must satisfy second-order (SOCs) and Kuhn-Tucker conditions

$$\max_{x} f(x)$$
  
subject to  
$$g(x) \le 0$$

$$\mathcal{L}(x,\lambda) = f(x) - \lambda g(x)$$

- In the case of optimization, f(x) is our likelihood function; x's are the parameters
- The first-order conditions (FOCs) tell us what the optimal x's are
- Also must satisfy second-order (SOCs) and Kuhn-Tucker conditions
- In this case, the SOCs involve looking at the bordered Hessian





• Let's go through an example of how to estimate an econometric model with JuMP

• Let's go through an example of how to estimate an econometric model with JuMP

• There are four basic components to any JuMP model:

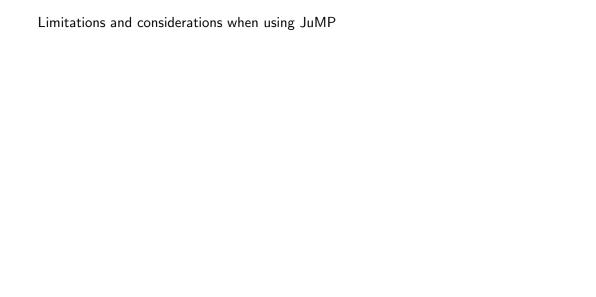
- Let's go through an example of how to estimate an econometric model with JuMP
- There are four basic components to any JuMP model:
  - 1. An optimizer

- Let's go through an example of how to estimate an econometric model with JuMP
- There are four basic components to any JuMP model:
  - 1. An optimizer
  - 2. Variables

- Let's go through an example of how to estimate an econometric model with JuMP
- There are four basic components to any JuMP model:
  - 1. An optimizer
  - 2. Variables
  - 3. Constraints

- Let's go through an example of how to estimate an econometric model with JuMP
- There are four basic components to any JuMP model:
  - 1. An optimizer
  - 2. Variables
  - 3. Constraints
  - 4. Objective function

- Let's go through an example of how to estimate an econometric model with JuMP
- There are four basic components to any JuMP model:
  - 1. An optimizer
  - 2. Variables
  - 3. Constraints
  - 4. Objective function
- This list is not too different from what goes into Optim.jl



- You cannot vectorize the objective function
  - i.e. everything needs to be expressed as a scalar

- You cannot vectorize the objective function
  - i.e. everything needs to be expressed as a scalar
- You cannot use Distributions. jl objects in the objective function

- You cannot vectorize the objective function
  - i.e. everything needs to be expressed as a scalar
- You cannot use Distributions.jl objects in the objective function
- It is a royal pain to extract the Hessian of the objective function
  - We need the Hessian to conduct statistical inference

- You cannot vectorize the objective function
  - i.e. everything needs to be expressed as a scalar
- You cannot use Distributions.jl objects in the objective function
- It is a royal pain to extract the Hessian of the objective function
  - We need the Hessian to conduct statistical inference
- It is very simple to add constraints

- You cannot vectorize the objective function
  - i.e. everything needs to be expressed as a scalar
- You cannot use Distributions.jl objects in the objective function
- It is a royal pain to extract the Hessian of the objective function
  - We need the Hessian to conduct statistical inference
- It is very simple to add constraints
- JuMP computes the Hessian of the Lagrangian when constraints are present

- You cannot vectorize the objective function
  - i.e. everything needs to be expressed as a scalar
- You cannot use Distributions.jl objects in the objective function
- It is a royal pain to extract the Hessian of the objective function
  - We need the Hessian to conduct statistical inference
- It is very simple to add constraints
- JuMP computes the Hessian of the Lagrangian when constraints are present
- This requires additional processing to obtain correct SEs under constraints