An Introduction to the Heterogeneous Agents Resources and toolKit

Chris Carroll, Matt White

Minicourse

"Hands-On Heterogeneous Agent Macroeconomics"

Goethe University and SAFE

2018-05-28

Agenda: A Flavor of HARK

1. "Microeconomic" models in HARK: the `AgentType` class

Agenda: A Flavor of HARK

1. "Microeconomic" models in HARK: the `AgentType` class
2. Example HARK model

Agenda: A Flavor of HARK

1. "Microeconomic" models in HARK: the `AgentType` class
2. Example HARK model
   - Consumption with permanent and transitory shocks to income

Agenda: A Flavor of HARK

1. “Microeconomic” models in HARK: the `AgentType` class
2. Example HARK model
   - Consumption with permanent and transitory shocks to income
3. 30,000 foot view: What else is in HARK?

Microeconomic Models in HARK

Microeconomic Models in HARK

- Concern decision-making of one agent

Microeconomic Models in HARK

- Concern decision-making of one agent
- Discrete time

Microeconomic Models in HARK

- Concern decision-making of one agent
- Discrete time
- Sequence of choices

Microeconomic Models in HARK

- Concern decision-making of one agent
- Discrete time
- Sequence of choices
  - Household: Consumption, labor supply, portfolio choice, etc

Microeconomic Models in HARK

- Concern decision-making of one agent
- Discrete time
- Sequence of choices
  - Household: Consumption, labor supply, portfolio choice, etc
  - Firm: Investment, Employment, R&D, ...

Microeconomic Models in HARK

- Concern decision-making of one agent
- Discrete time
- Sequence of choices
  - Household: Consumption, labor supply, portfolio choice, etc
  - Firm: Investment, Employment, R&D, ...
- Agents treat inputs to problem as *exogenous*

Microeconomic Models in HARK

- Concern decision-making of one agent
- Discrete time
- Sequence of choices
  - Household: Consumption, labor supply, portfolio choice, etc
  - Firm: Investment, Employment, R&D, ...
- Agents treat inputs to problem as *exogenous*

Key restriction: Essentially, Bellman equation

Microeconomic Models in HARK

- Concern decision-making of one agent
- Discrete time
- Sequence of choices
  - Household: Consumption, labor supply, portfolio choice, etc
  - Firm: Investment, Employment, R&D, ...
- Agents treat inputs to problem as *exogenous*

Key restriction: Essentially, Bellman equation

Model solution can be constructed as iteration on sequence of "one period problems," conditional on solution to subsequent period.

Two kinds of heterogeneity

- *Ex post* heterogeneity: Agents differ because a different sequence of events or shocks has happened to them

Two kinds of heterogeneity

- *Ex post* heterogeneity: Agents differ because a different sequence of events or shocks has happened to them
  - Luck of the draw

Two kinds of heterogeneity

- *Ex post* heterogeneity: Agents differ because a different sequence of events or shocks has happened to them
  - Luck of the draw
- *Ex ante* heterogeneity: Agents differ in objectives, preferences, expectations, etc before anything "happens" to them

Two kinds of heterogeneity

- *Ex post* heterogeneity: Agents differ because a different sequence of events or shocks has happened to them
  - Luck of the draw
- *Ex ante* heterogeneity: Agents differ in objectives, preferences, expectations, etc before anything "happens" to them
  - Some people are more risk averse than others, e.g.

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
    - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

  $\Rightarrow$ conceptual (and operational) groupings of similar things

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

  $\Rightarrow$ conceptual (and operational) groupings of similar things
  - Start by defining needed items as a *class*

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

  $\Rightarrow$ conceptual (and operational) groupings of similar things
  - Start by defining needed items as a *class*
    - Example of a "class": `PerfForesightConsumerType`

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

  $\Rightarrow$ conceptual (and operational) groupings of similar things
  - Start by defining needed items as a *class*
    - Example of a "class": `PerfForesightConsumerType`
  - **Attributes**: What characteristics define one *instance*?

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

  $\Rightarrow$ conceptual (and operational) groupings of similar things
  - Start by defining needed items as a *class*
    - Example of a "class": `PerfForesightConsumerType`
  - **Attributes**: What characteristics define one *instance*?
    - e.g., an instance of a `PerfForesightConsumerType` with $\{\mathrm{R}, \beta, ...\} = \{1.05, 1/1.05, ...\}$

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

  $\Rightarrow$ conceptual (and operational) groupings of similar things
  - Start by defining needed items as a *class*
    - Example of a "class": `PerfForesightConsumerType`
  - **Attributes**: What characteristics define one *instance*?
    - e.g., an instance of a `PerfForesightConsumerType` with $\{\mathrm{R}, \beta, ...\} = \{1.05, 1/1.05, ...\}$
  - **Methods**: What actions can instances of the class "do"?

Crash Course in Object-Oriented Programming

(For more, see WhyPython)

- Traditional programming is *procedural*:
  - Accomplish task step by step; intuitive
- Python is an *object-oriented* language:

  $\Rightarrow$ conceptual (and operational) groupings of similar things
  - Start by defining needed items as a *class*
    - Example of a "class": `PerfForesightConsumerType`
  - **Attributes**: What characteristics define one *instance*?
    - e.g., an instance of a `PerfForesightConsumerType` with $\{\mathsf{R}, \beta, ...\} = \{1.05, 1/1.05, ...\}$
  - **Methods**: What actions can instances of the class "do"?
    - e.g., calculate 'impatience' implied by its parameter values

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`
  - e.g. `PerfForesightConsumerType` is an `AgentType` subclass

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`
  - e.g. `PerfForesightConsumerType` is an `AgentType` subclass
  - Includes attributes, functions, and methods...

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`
    - e.g. `PerfForesightConsumerType` is an `AgentType` subclass
    - Includes attributes, functions, and methods...
        - All `AgentType` subclasses have a `solve()` method

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`
  - e.g. `PerfForesightConsumerType` is an `AgentType` subclass
  - Includes attributes, functions, and methods...
    - All `AgentType` subclasses have a `solve()` method
  - Common structure $\Rightarrow$ different models "play nicely" together

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`
    - e.g. `PerfForesightConsumerType` is an `AgentType` subclass
    - Includes attributes, functions, and methods...
        - All `AgentType` subclasses have a `solve()` method
    - Common structure $\Rightarrow$ different models "play nicely" together
    - Even though guts of `solve()` method differ for each subclass

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`
  - e.g. `PerfForesightConsumerType` is an `AgentType` subclass
  - Includes attributes, functions, and methods...
    - All `AgentType` subclasses have a `solve()` method
  - Common structure ⇒ different models "play nicely" together
  - Even though guts of `solve()` method differ for each subclass
  - Much easier to compare and exchange models

HARK's "Master Class": `AgentType`

- General purpose class for representing economic agents
- Each model creates a subclass of `AgentType`
  - e.g. `PerfForesightConsumerType` is an `AgentType` subclass
  - Includes attributes, functions, and methods...
    - All `AgentType` subclasses have a `solve()` method
  - Common structure $\Rightarrow$ different models "play nicely" together
  - Even though guts of `solve()` method differ for each subclass
  - Much easier to compare and exchange models
- Complex models extend basic ones through "class inheritance"

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`
- Define Information needed to solve one period back

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`
- Define Information needed to solve one period back
  - Variables, Distributions, Parameters, ...

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`
- Define Information needed to solve one period back
  - Variables, Distributions, Parameters, ...
- Keep Going ...

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`
- Define Information needed to solve one period back
  - Variables, Distributions, Parameters, ...
- Keep Going ...
  - Life Cycle? To birth

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`
- Define Information needed to solve one period back
  - Variables, Distributions, Parameters, ...
- Keep Going ...
  - Life Cycle? To birth
  - Infinite Horizon? To convergence

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`
- Define Information needed to solve one period back
  - Variables, Distributions, Parameters, ...
- Keep Going ...
  - Life Cycle? To birth
  - Infinite Horizon? To convergence
  - Infinite Horizon with Seasons? To seasonal convergence

Defining An Agent's Problem

- Define Behavior in some "Terminal Period"
  - Created by the class's method `solveTerminal`
  - Example that can be used for finite or infinite horizon:
    - `solution_terminal.cFunc = consume everything`
    - `solution_terminal.vPfunc = u'(consume everything)`
- Define Information needed to solve one period back
  - Variables, Distributions, Parameters, ...
- Keep Going ...
  - Life Cycle? To birth
  - Infinite Horizon? To convergence
  - Infinite Horizon with Seasons? To seasonal convergence
  - ...

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`
  - `IncomeDstn = [too much to put here]`

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`
  - `IncomeDstn = [too much to put here]`
  - Life Cycle version:

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`
  - `IncomeDstn = [too much to put here]`
  - Life Cycle version:
    - `PermGroFac = [1.005,...,1.005,0.4,0.998,...,0.998]`

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`
  - `IncomeDstn = [too much to put here]`
  - Life Cycle version:
    - `PermGroFac = [1.005,...,1.005,0.4,0.998,...,0.998]`
    - `LivPrb = [1,...,1,0.997,0.994,0.991,...,0]`

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`
  - `IncomeDstn = [too much to put here]`
  - Life Cycle version:
    - `PermGroFac = [1.005,...,1.005,0.4,0.998,...,0.998]`
    - `LivPrb = [1,...,1,0.997,0.994,0.991,...,0]`
  - Infinite Horizon version:

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`
  - `IncomeDstn = [too much to put here]`
  - Life Cycle version:
    - `PermGroFac = [1.005,...,1.005,0.4,0.998,...,0.998]`
    - `LivPrb = [1,...,1,0.997,0.994,0.991,...,0]`
  - Infinite Horizon version:
    - `PermGroFac = [1.02]`

Defining An Agent's Problem

- Example: `IndShockConsumerType` consumption model:
  - `CRRA = 3.2`
  - `DiscFac = 0.96`
  - `Rfree = 1.03`
  - `IncomeDstn = [too much to put here]`
  - Life Cycle version:
    - `PermGroFac = [1.005,...,1.005,0.4,0.998,...,0.998]`
    - `LivPrb = [1,...,1,0.997,0.994,0.991,...,0]`
  - Infinite Horizon version:
    - `PermGroFac = [1.02]`
    - `LivPrb = [0.98]`

Workhorse: Buffer Stock Consumption Model

Class `IndShockConsumerType`

- Inherits attributes of `PerfForesightConsumerType`

Workhorse: Buffer Stock Consumption Model

Class `IndShockConsumerType`

- Inherits attributes of `PerfForesightConsumerType`
  - Geometric discounting β per period

Workhorse: Buffer Stock Consumption Model

Class `IndShockConsumerType`

- Inherits attributes of `PerfForesightConsumerType`
  - Geometric discounting β per period
  - One choice: How much to consume vs save

Workhorse: Buffer Stock Consumption Model

Class `IndShockConsumerType`

- Inherits attributes of `PerfForesightConsumerType`
  - Geometric discounting β per period
  - One choice: How much to consume vs save
  - CRRA utility from consumption

Workhorse: Buffer Stock Consumption Model

Class `IndShockConsumerType`

- Inherits attributes of `PerfForesightConsumerType`
  - Geometric discounting β per period
  - One choice: How much to consume vs save
  - CRRA utility from consumption
  - Exogenous interest factor for asset returns

Workhorse: Buffer Stock Consumption Model

Class `IndShockConsumerType`

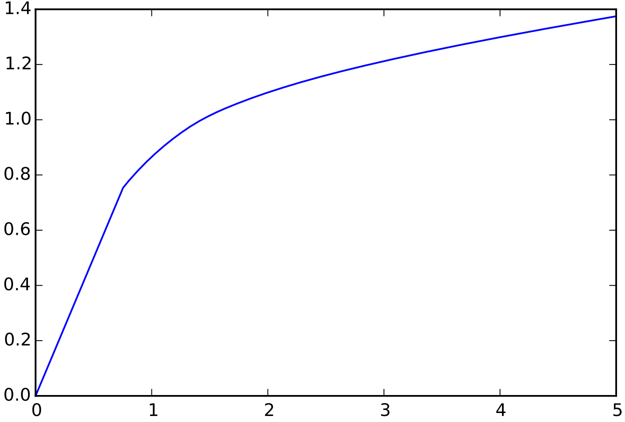
- Inherits attributes of `PerfForesightConsumerType`
  - Geometric discounting β per period
  - One choice: How much to consume vs save
  - CRRA utility from consumption
  - Exogenous interest factor for asset returns
- Adds assumptions about income uncertainy and constraints

Workhorse: Buffer Stock Consumption Model

Class `IndShockConsumerType`

- Inherits attributes of `PerfForesightConsumerType`
  - Geometric discounting β per period
  - One choice: How much to consume vs save
  - CRRA utility from consumption
  - Exogenous interest factor for asset returns
- Adds assumptions about income uncertainty and constraints
  - Mathematical Details: Formal model

Buffer Stock Model Consumption Function



Horizontal Axis: "Money"; Vertical Axis: "Spending"

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for "macroeconomic" models: `Market` class

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for "macroeconomic" models: `Market` class
- Several extensions of basic consumption-saving model

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for “macroeconomic” models: `Market` class
- Several extensions of basic consumption-saving model
- Some small demonstration exercises

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for "macroeconomic" models: `Market` class
- Several extensions of basic consumption-saving model
- Some small demonstration exercises
- All results from several papers:

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for "macroeconomic" models: `Market` class
- Several extensions of basic consumption-saving model
- Some small demonstration exercises
- All results from several papers:
  - "The Distribution of Wealth and the Marginal Propensity to Consume" by Carroll, Slacalek, Tokuoka, and White (2017)

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for "macroeconomic" models: `Market` class
- Several extensions of basic consumption-saving model
- Some small demonstration exercises
- All results from several papers:
  - "The Distribution of Wealth and the Marginal Propensity to Consume" by Carroll, Slacalek, Tokuoka, and White (2017)
  - "Sticky Expectations and Consumption Dynamics" by Carroll, Crawley, Slacalek, Tokuoka, and White (2018)

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for "macroeconomic" models: `Market` class
- Several extensions of basic consumption-saving model
- Some small demonstration exercises
- All results from several papers:
  - "The Distribution of Wealth and the Marginal Propensity to Consume" by Carroll, Slacalek, Tokuoka, and White (2017)
  - "Sticky Expectations and Consumption Dynamics" by Carroll, Crawley, Slacalek, Tokuoka, and White (2018)
  - Several others are close

What Else Is In HARK or the Econ-ARK?

- General purpose tools for generating and representing distributions, interpolated functions, etc
- Tools for estimation / optimization (fairly sparse)
- Framework for "macroeconomic" models: `Market` class
- Several extensions of basic consumption-saving model
- Some small demonstration exercises
- All results from several papers:
  - "The Distribution of Wealth and the Marginal Propensity to Consume" by Carroll, Slacalek, Tokuoka, and White (2017)
  - "Sticky Expectations and Consumption Dynamics" by Carroll, Crawley, Slacalek, Tokuoka, and White (2018)
  - Several others are close
- Much room for improvement: endogenous labor supply (e.g.)

Other Consumption-Saving Models in HARK

- `TractableBufferStock`: Highly specialized idiosync shocks

And even more to come...

Other Consumption-Saving Models in HARK

- `TractableBufferStock`: Highly specialized idiosync shocks
- `MarkovModel`: $\cancel{D}, \Gamma, F_\theta, F_\psi, R$ vary by discrete state

And even more to come...

Other Consumption-Saving Models in HARK

- `TractableBufferStock`: Highly specialized idiosync shocks
- `MarkovModel`: $\cancel{D}, \Gamma, F_\theta, F_\psi, R$ vary by discrete state
- `ExplicitPermInc`: Same as `IndShock`, but not normalized
- `PersistentShock`: "Permanent" shocks not fully permanent

And even more to come...

Other Consumption-Saving Models in HARK

- `TractableBufferStock`: Highly specialized idiosync shocks
- `MarkovModel`: $\cancel{D}, \Gamma, F_\theta, F_\psi, R$ vary by discrete state
- `ExplicitPermInc`: Same as `IndShock`, but not normalized
- `PersistentShock`: "Permanent" shocks not fully permanent
- `MedShock`: 2nd cons good with random marginal utility

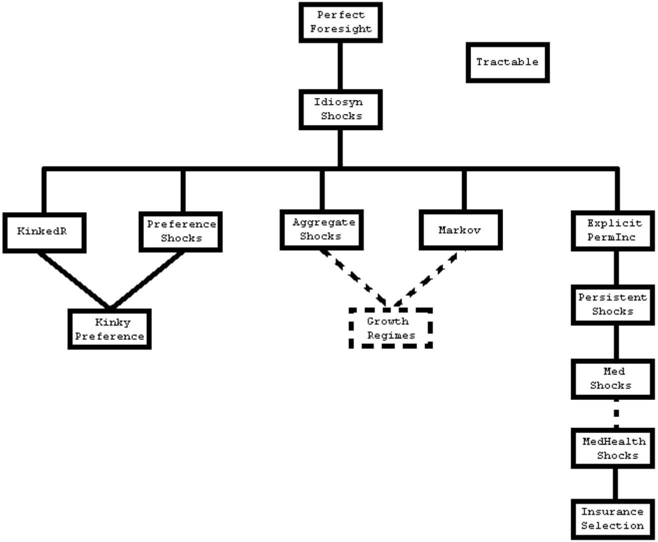
And even more to come...

Other Consumption-Saving Models in HARK

- `TractableBufferStock`: Highly specialized idiosync shocks
- `MarkovModel`: $\cancel{D}, \Gamma, F_\theta, F_\psi, R$ vary by discrete state
- `ExplicitPermInc`: Same as `IndShock`, but not normalized
- `PersistentShock`: "Permanent" shocks not fully permanent
- `MedShock`: 2nd cons good with random marginal utility
- `MedHealthShock`:* Medical shocks plus discrete health states
- `DynInsSel`:* ...plus choice over medical insurance contracts

And even more to come...

Consumption-Saving Model Tree



Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link

Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link
- "Macroeconomic" framework and models Link

Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link
- "Macroeconomic" framework and models Link
- To do: endogenous labor supply models Link

Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link
- "Macroeconomic" framework and models Link
- To do: endogenous labor supply models Link
- To do: durable goods models Link

Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link
- "Macroeconomic" framework and models Link
- To do: endogenous labor supply models Link
- To do: durable goods models Link
- To do: various bits, large and small Link

Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link

Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link
- "Macroeconomic" framework and models Link

Topics for Further Discussion

Time is short, but I could talk about...

- Class inheritance / model recombination Link
- “Macroeconomic” framework and models Link
- Roadmap Link

Planned Future of Additions to HARK

1. Endogenous labor supply models Link

Planned Future of Additions to HARK

1. Endogenous labor supply models Link
2. Durable goods models Link

Planned Future of Additions to HARK

1. Endogenous labor supply models Link
2. Durable goods models Link
3. Various bits, large and small Link

The Future of HARK: Incorporating Labor (1/4)

Model of labor supply on intensive margin:

$$u(c,\ell) = ((1-\ell)^{\alpha} c)^{1-\rho}/(1-\rho),$$

$$v_t(b_t,\theta_t) = \max_{c_t,\ell_t} u(c_t,\ell_t) + \beta \cancel{D}_t \mathbb{E}_t \left[(\psi_{t+1}\Gamma_t)^{1-\rho} v_{t+1}(b_{t+1},\theta_{t+1}) \right] \text{ s.t.}$$

$$y_t = \ell_t \theta_t, \qquad \ell_t \in [0,1],$$

$$a_t = m_t + y_t - c_t, \qquad a_t \geq \underline{a},$$

$$b_{t+1} = R/(\Gamma_t \psi_{t+1}) a_t,$$

$$\psi_{t+1} \sim F_{\psi t+1}(\psi), \qquad \theta_{t+1} \sim F_{\theta t+1}(\theta), \quad \mathbb{E}[\psi_{t+1}] = 1.$$

The Future of HARK: Incorporating Labor (2/4)

Model of labor supply on extensive margin:

$$u(c, \ell) = c^{1-\rho}/(1-\rho) - \alpha\ell,$$

$$v_t(b_t, \theta_t, \ell_{t-1}) = \max_{c_t, \ell_t} u(c_t, \ell_t) + \beta \cancel{D}_t \mathbb{E}_t \left[(\psi_{t+1}\Gamma_t)^{1-\rho} v_{t+1}(b_{t+1}, \theta_{t+1}, \ell_t) \right]$$

$$y_t = \ell_t \theta_t, \qquad \ell_t \in \{0, \ell_{t-1}\},$$

$$a_t = m_t + y_t - c_t, \quad a_t \geq \underline{a},$$

$$b_{t+1} = R/(\Gamma_t \psi_{t+1}) a_t,$$

$$\psi_{t+1} \sim F_{\psi t+1}(\psi), \quad \theta_{t+1} \sim F_{\theta t+1}(\theta), \;\; \mathbb{E}[\psi_{t+1}] = 1.$$

The Future of HARK: Incorporating Labor (3/4)

Model of endogenous employment search:

$$u(c,s) = ((1-s)^{\alpha} c)^{1-\rho}/(1-\rho),$$

$$v_t(m_t,e_t) = \max_{c_t,s_t} u(c_t,s_t) + \beta \cancel{D}_t \mathbb{E}_t \left[(\psi_{t+1}\Gamma_t)^{1-\rho} v_{t+1}(m_{t+1},e_{t+1}) \right] \text{ s.t.}$$

$$a_t = m_t - c_t, \quad a_t \geq \underline{a}, \quad s_t \in [0,1],$$

$$m_{t+1} = R/(\Gamma_t^e \psi_{t+1}) a_t + \theta_t e_{t+1} + \underline{b}(1-e_{t+1}),$$

$$\text{Prob}(e_{t+1}=1 | e_t=0) = s_t, \qquad \text{Prob}(e_{t+1}=0 | e_t=1) = \mho,$$

$$\psi_{t+1} \sim F^e_{\psi t+1}(\psi), \quad \theta_{t+1} \sim F_{\theta t+1}(\theta), \quad \mathbb{E}[\psi_t] = 1.$$

The Future of HARK: Incorporating Labor (4/4)

Applications of `Market` for labor models:

- Non-trivial calculation of $L_t = \int_0^1 \ell_{it} p_{it} \theta_{it} di$ for Cobb-Douglas

Back

The Future of HARK: Incorporating Labor (4/4)

Applications of `Market` for labor models:

- Non-trivial calculation of $L_t = \int_0^1 \ell_{it} p_{it} \theta_{it} di$ for Cobb-Douglas
- Disutility of employment search and probability of job loss depend on labor market slackness

Back

The Future of HARK: Incorporating Labor (4/4)

Applications of `Market` for labor models:

- Non-trivial calculation of $L_t = \int_0^1 \ell_{it} p_{it} \theta_{it} di$ for Cobb-Douglas
- Disutility of employment search and probability of job loss depend on labor market slackness
- Can look at behavior in response to change in SS, etc

Back

The Future of HARK: Durable Goods (1/3)

General durable goods model:

$$u(c, d) = (c^{\alpha}, d^{1-\alpha})^{1-\rho}/(1-\rho).$$

$$v_t(m_t, d_t) = \max_{c_t, i_t} u(c_t, d_t) + \beta \cancel{D}_t \mathbb{E}_t \left[(\psi_{t+1}\Gamma_t)^{1-\rho} v_{t+1}(m_{t+1}, d_{t+1}) \right] \text{ s.t.}$$

$$a_t = m_t - c_t, \qquad a_t \geq \underline{a},$$

$$D_t = d_t + g(i_t), \qquad d_{t+1} = (1-\delta_{t+1})D_t, \qquad \delta_{t+1} \sim F_{\delta}(\delta),$$

$$m_{t+1} = R/(\Gamma_t \psi_{t+1}) a_t + \theta_{t+1},$$

$$\psi_{t+1} \sim F_{\psi t+1}(\psi), \qquad \theta_{t+1} \sim F_{\theta t+1}(\theta), \quad \mathbb{E}[\psi_t] = 1.$$

The Future of HARK: Durable Goods (2/3)

Variations of durable goods model require different solvers:

- Easiest case: $g(i_t)$ is concave, $i_t \in \mathrm{R}$. Every end-of-period state (a_t, D_t) associated with *some* beginning of period state.
- Slightly harder: $i_t \geq 0$, must handle constraint

The Future of HARK: Durable Goods (2/3)

Variations of durable goods model require different solvers:

- Easiest case: $g(i_t)$ is concave, $i_t \in \mathrm{R}$. Every end-of-period state (a_t, D_t) associated with *some* beginning of period state.
- Slightly harder: $i_t \geq 0$, must handle constraint
- Somewhat harder: $g(i_t) = \pi i_t$. One locus in (a_t, D_t) space is optimal; each point on optimal (a_t, D_t) locus associated with locus in (m_t, d_t) space.

The Future of HARK: Durable Goods (2/3)

Variations of durable goods model require different solvers:

- Easiest case: $g(i_t)$ is concave, $i_t \in \mathrm{R}$. Every end-of-period state (a_t, D_t) associated with *some* beginning of period state.
- Slightly harder: $i_t \geq 0$, must handle constraint
- Somewhat harder: $g(i_t) = \pi i_t$. One locus in (a_t, D_t) space is optimal; each point on optimal (a_t, D_t) locus associated with locus in (m_t, d_t) space.
- Even harder: $g(i_t) = \widehat{g}(i_t) + K\mathbf{1}(i_t \neq 0)$, with $\widehat{g}(0) = 0$ and $\widehat{g}(\cdot)$ concave. Must check $i_t = 0$ soln everywhere, discont.

The Future of HARK: Durable Goods (2/3)

Variations of durable goods model require different solvers:

- Easiest case: $g(i_t)$ is concave, $i_t \in \mathrm{R}$. Every end-of-period state (a_t, D_t) associated with *some* beginning of period state.
- Slightly harder: $i_t \geq 0$, must handle constraint
- Somewhat harder: $g(i_t) = \pi i_t$. One locus in (a_t, D_t) space is optimal; each point on optimal (a_t, D_t) locus associated with locus in (m_t, d_t) space.
- Even harder: $g(i_t) = \widehat{g}(i_t) + K\mathbf{1}(i_t \neq 0)$, with $\widehat{g}(0) = 0$ and $\widehat{g}(\cdot)$ concave. Must check $i_t = 0$ soln everywhere, discont.
- Just ugh: $g(i_t) = \pi i_t + K\mathbf{1}(i_t \neq 0)$, $i_t \geq 0$.

The Future of HARK: Durable Goods (3/3)

Applications for `Market` with durable goods:

- Endogenous pricing of durable good: housing market

Back

The Future of HARK: Durable Goods (3/3)

Applications for `Market` with durable goods:

- Endogenous pricing of durable good: housing market
- Dynamics of demand for durables after an aggregate shock

Back

The Future of HARK: Durable Goods (3/3)

Applications for `Market` with durable goods:

- Endogenous pricing of durable good: housing market
- Dynamics of demand for durables after an aggregate shock
- Some specifications overlap with health models

Back

The Future of HARK: Small To-Do Items

Contributions that would get your feet wet in HARK:

- Extend perfect foresight to handle borrowing constraint
- Bequest motives: warm glow, other?

The Future of HARK: Small To-Do Items

Contributions that would get your feet wet in HARK:

- Extend perfect foresight to handle borrowing constraint
- Bequest motives: warm glow, other?
- Fix/improve aggregate shocks model: handle life cycle models
- Fix/generalize `ExplicitPermInc` models: `PermGroFunc`
- Portfolio allocation models; eventually: asset pricing

The Future of HARK: Small To-Do Items

Contributions that would get your feet wet in HARK:

- Extend perfect foresight to handle borrowing constraint
- Bequest motives: warm glow, other?
- Fix/improve aggregate shocks model: handle life cycle models
- Fix/generalize `ExplicitPermInc` models: `PermGroFunc`
- Portfolio allocation models; eventually: asset pricing
- Advanced features on more solvers: cubic spline interpolation
- Various numeric methods detached from particular models

The Future of HARK: Heavy Lifting

If you're feeling ambitious or are comfortable with HARK:

- Incorporate `opencl4py` with basic consumption-saving model. "Repack" model inputs into memory buffers, pass to OpenCL solver. OpenCL simulator: easier, big gains for some models.

Back

The Future of HARK: Heavy Lifting

If you're feeling ambitious or are comfortable with HARK:

- Incorporate `opencl4py` with basic consumption-saving model. "Repack" model inputs into memory buffers, pass to OpenCL solver. OpenCL simulator: easier, big gains for some models.
- Aggregate shocks with explicit permanent income. One endogenous state variable, two exogenous state variables. Future candidate for GPU computing.

Back

The Future of HARK: Heavy Lifting

If you're feeling ambitious or are comfortable with HARK:

- Incorporate `opencl4py` with basic consumption-saving model. "Repack" model inputs into memory buffers, pass to OpenCL solver. OpenCL simulator: easier, big gains for some models.
- Aggregate shocks with explicit permanent income. One endogenous state variable, two exogenous state variables. Future candidate for GPU computing.
- Generalized Markov solver: make "solution schema" so that Markov state can be added to any correctly specified solver

Back

The Future of HARK: Heavy Lifting

If you're feeling ambitious or are comfortable with HARK:

- Incorporate `opencl4py` with basic consumption-saving model. "Repack" model inputs into memory buffers, pass to OpenCL solver. OpenCL simulator: easier, big gains for some models.
- Aggregate shocks with explicit permanent income. One endogenous state variable, two exogenous state variables. Future candidate for GPU computing.
- Generalized Markov solver: make "solution schema" so that Markov state can be added to any correctly specified solver
- Models of firm creation / bankruptcy / investment / hiring

Back

Example Model: Basic Consumption-Saving

Consumption-saving model with idiosyncratic permanent and transitory shocks to income (normalized format):

$$u(c) = c^{1-\rho}/(1-\rho).$$

$$v_t(m_t) = \max_{c_t} u(c_t) + \beta \cancel{D}_t \mathbb{E}_t \left[(\psi_{t+1}\Gamma_{t+1})^{1-\rho} v_{t+1}(m_{t+1}) \right] \text{ s.t.}$$

$$a_t = m_t - c_t, \quad a_t \geq \underline{a},$$

$$m_{t+1} = R/(\Gamma_{t+1}\psi_{t+1})a_t + \theta_{t+1},$$

$$\psi_{t+1} \sim F_{\psi t+1}(\psi), \quad \theta_{t+1} \sim F_{\theta t+1}(\theta), \quad \mathbb{E}[\psi_t] = 1.$$

Example Model: Basic Consumption-Saving

Model solution in two lines:

$$\text{FOC: } u'(c_t) = R\beta \cancel{D}\mathbb{E}_t\left[(\psi_{t+1}\Gamma_{t+1})^{-\rho} v'_{t+1}(m_{t+1})\right],$$

$$\text{EC: } v'_t(m_t) = u'(c_t).$$

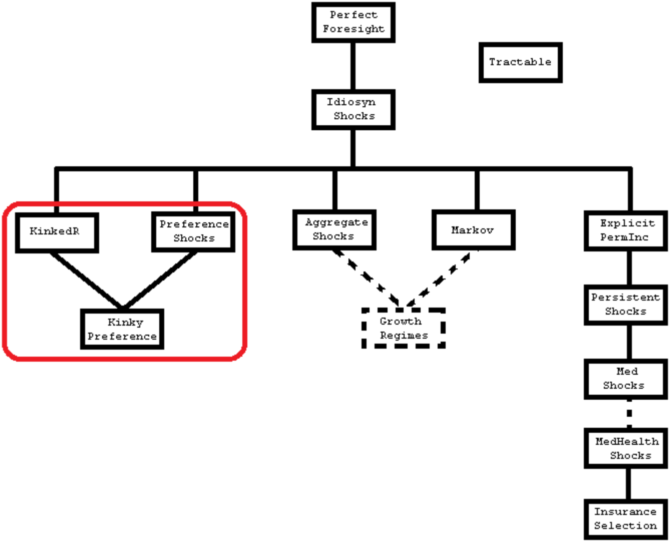
Will use endogenous grid method:

$$\mathfrak{v}'_t(a_t) \equiv R\beta\mathbb{E}_t\left[(\psi_{t+1}\Gamma_{t+1})^{-\rho} v'_{t+1}(m_{t+1})|a_t\right],$$

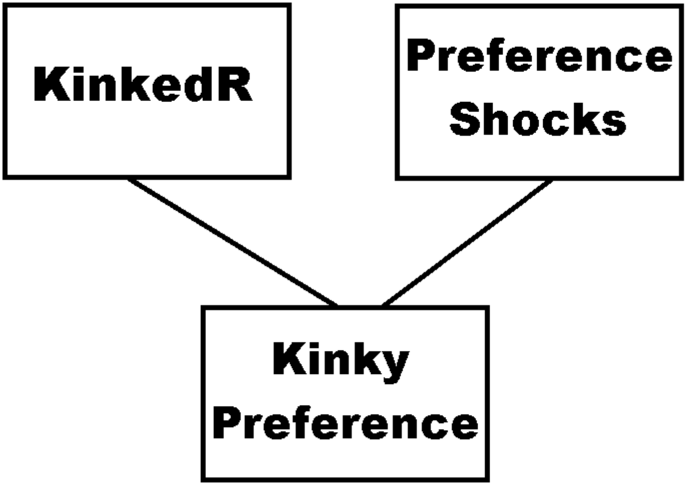
$$c_t = \mathfrak{v}'_t(a_t)^{-\rho}, \qquad m_t = a_t + c_t \text{ (for exogenous set of } \{a_t\}).$$

Back

Consumption-Saving Model Tree



Consumption-Saving Model Tree



Kinked R: Costly Borrowing (1/3)

Make one small adjustment to idiosyncratic income shocks model: interest rate on borrowing is higher than rate on saving.

$$
\begin{aligned}
u(c) &= \frac{c^{1-\rho}}{1-\rho}, \\
v(m_t) &= \max_{c_t} u(c_t) + \beta \cancel{D}_{t+1} \mathbb{E}[v_{t+1}(m_{t+1})], \\
a_t &= m_t - c_t, \qquad a_t \geq \underline{a}, \\
m_{t+1} &= R/(\Gamma_{t+1}\psi_{t+1}) a_t + \theta_{t+1}, \\
\theta_{t+1} \sim F_{\theta t+1}, & \qquad \psi_{t+1} \sim F_{\psi t+1}, \quad \mathbb{E}[\psi_{t+1}] = 1, \\
R &= \begin{cases} R_{boro} & \text{if } a_t < 0 \\ R_{save} & \text{if } a_t > 0 \end{cases}, \qquad R_{boro} \geq R_{save}.
\end{aligned}
$$

Kinked R: Costly Borrowing (2/3)

`ConsKinkedRsolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

Kinked R: Costly Borrowing (2/3)

`ConsKinkedRsolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- Store new attributes `Rboro` and `Rsave`

Kinked R: Costly Borrowing (2/3)

`ConsKinkedRsolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- Store new attributes `Rboro` and `Rsave`

Additions to `prepareToCalcEndOfPrdvP`:

Kinked R: Costly Borrowing (2/3)

`ConsKinkedRsolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- Store new attributes `Rboro` and `Rsave`

Additions to `prepareToCalcEndOfPrdvP`:

- Four lines to use correct value of R for each value of a_t

Kinked R: Costly Borrowing (2/3)

`ConsKinkedRsolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- Store new attributes `Rboro` and `Rsave`

Additions to `prepareToCalcEndOfPrdvP`:

- Four lines to use correct value of R for each value of a_t
- One line to apply that change to calculation of m_{t+1}

Kinked R: Costly Borrowing (2/3)

`ConsKinkedRsolver` inherits from `ConsIndShockSolver`

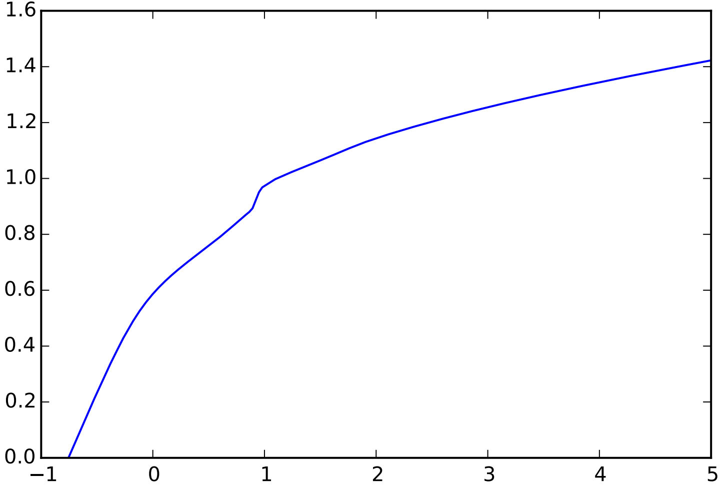
Additions to `__init__` method:

- Store new attributes `Rboro` and `Rsave`

Additions to `prepareToCalcEndOfPrdvP`:

- Four lines to use correct value of R for each value of a_t
- One line to apply that change to calculation of m_{t+1}
- Three lines to recalculate minimum MPC and human wealth

Kinked R: Costly Borrowing (3/3)



Marginal Utility Shocks (1/4)

Consider another small modification to `IndShockModel`:

- Multiplicative (idiosyncratic) shocks to utility each period.

$$
\begin{aligned}
u(c;\eta) &= \eta\frac{c^{1-\rho}}{1-\rho}, \qquad \eta_t \sim F_\eta, \\
v(m_t,\eta_t) &= \max_{c_t} u(c_t;\eta_t) + \beta \not{D}_{t+1}\mathbb{E}[v_{t+1}(m_{t+1})], \\
a_t &= m_t - c_t, \qquad a_t \geq \underline{a}, \\
m_{t+1} &= R/(\Gamma_{t+1}\psi_{t+1})a_t + \theta_{t+1}, \\
\theta_{t+1} \sim F_{\theta t+1}, &\qquad \psi_{t+1} \sim F_{\psi t+1}, \quad \mathbb{E}[\psi_{t+1}] = 1.
\end{aligned}
$$

Marginal Utility Shocks (1/4)

Consider another small modification to `IndShockModel`:

- Multiplicative (idiosyncratic) shocks to utility each period.
- Consumption "more valuable" in some periods than others.

$$
\begin{aligned}
u(c;\eta) \quad &= \quad \eta \frac{c^{1-\rho}}{1-\rho}, \qquad \eta_t \sim F_\eta, \\
v(m_t,\eta_t) \quad &= \quad \max_{c_t} u(c_t;\eta_t) + \beta \not{D}_{t+1} \mathbb{E}[v_{t+1}(m_{t+1})], \\
a_t \quad &= \quad m_t - c_t, \qquad a_t \geq \underline{a}, \\
m_{t+1} \quad &= \quad R/(\Gamma_{t+1}\psi_{t+1})a_t + \theta_{t+1}, \\
\theta_{t+1} \sim F_{\theta t+1}, \quad & \quad\quad \psi_{t+1} \sim F_{\psi t+1}, \quad \mathbb{E}[\psi_{t+1}] = 1.
\end{aligned}
$$

Marginal Utility Shocks (2/4)

New input `PrefShkDstn` is constructed:

- `PrefShkStd`: Standard deviation of (log) pref shocks

Marginal Utility Shocks (2/4)

New input `PrefShkDstn` is constructed:

- `PrefShkStd`: Standard deviation of (log) pref shocks
- `PrefShkCount`: Number of discrete shocks in "body"

Marginal Utility Shocks (2/4)

New input `PrefShkDstn` is constructed:

- `PrefShkStd`: Standard deviation of (log) pref shocks
- `PrefShkCount`: Number of discrete shocks in "body"
- `PrefShkTailCount`: Discrete shocks in "augmented tail"

Marginal Utility Shocks (3/4)

`ConsPrefShockSolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

Marginal Utility Shocks (3/4)

`ConsPrefShockSolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- 2 lines: Store preference shock distribution `PrefShkDstn`

Marginal Utility Shocks (3/4)

`ConsPrefShockSolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- 2 lines: Store preference shock distribution `PrefShkDstn`

Replace `getPointsForInterpolation`

Marginal Utility Shocks (3/4)

`ConsPrefShockSolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- 2 lines: Store preference shock distribution `PrefShkDstn`

Replace `getPointsForInterpolation`

- 8 lines: Values of c_t and m_t for each η_t in `PrefShkDstn`

Marginal Utility Shocks (3/4)

`ConsPrefShockSolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- 2 lines: Store preference shock distribution `PrefShkDstn`

Replace `getPointsForInterpolation`

- 8 lines: Values of c_t and m_t for each η_t in `PrefShkDstn`

Replace `usePointsForInterpolation`

Marginal Utility Shocks (3/4)

`ConsPrefShockSolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- 2 lines: Store preference shock distribution `PrefShkDstn`

Replace `getPointsForInterpolation`

- 8 lines: Values of c_t and m_t for each η_t in `PrefShkDstn`

Replace `usePointsForInterpolation`

- 6 lines: Construct `cFunc` as a `LinearInterpOnInterp1D`

Marginal Utility Shocks (3/4)

`ConsPrefShockSolver` inherits from `ConsIndShockSolver`

Additions to `__init__` method:

- 2 lines: Store preference shock distribution `PrefShkDstn`

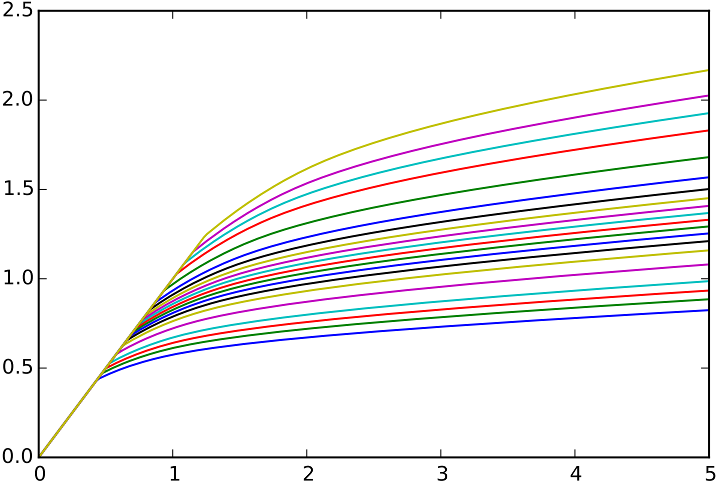
Replace `getPointsForInterpolation`

- 8 lines: Values of c_t and m_t for each η_t in `PrefShkDstn`

Replace `usePointsForInterpolation`

- 6 lines: Construct `cFunc` as a `LinearInterpOnInterp1D`
- 6 lines: Make `vPfunc` by integrating marginal utility across η_t

Marginal Utility Shocks (4/4)



Combination Inheritance: “Kinky Preferences” (1/4)

Combine those two extensions to `IndShockModel`:

- Borrowing has higher interest rate than saving...

Combination Inheritance: "Kinky Preferences" (1/4)

Combine those two extensions to `IndShockModel`:

- Borrowing has higher interest rate than saving...
- ...and there are shocks to marginal utility

Combination Inheritance: “Kinky Preferences” (1/4)

Combine those two extensions to `IndShockModel`:

- Borrowing has higher interest rate than saving...
- ...and there are shocks to marginal utility
- HARK makes this pretty easy

Combination Inheritance: "Kinky Preferences" (2/4)

$$
\begin{aligned}
u(c,\eta) &= \eta\frac{c^{1-\rho}}{1-\rho}, \qquad \eta_t \sim F_\eta, \\
v(m_t,\eta_t) &= \max_{c_t} u(c_t) + \beta \cancel{D}_{t+1}\mathbb{E}[v_{t+1}(m_{t+1})], \\
a_t &= m_t - c_t, \qquad a_t \geq \underline{a}, \\
m_{t+1} &= R/(\Gamma_{t+1}\psi_{t+1})a_t + \theta_{t+1}, \\
\theta_{t+1} \sim F_{\theta t+1}, & \quad \psi_{t+1} \sim F_{\psi t+1}, \quad \mathbb{E}[\psi_{t+1}] = 1, \\
R &= \begin{cases} R_{boro} & \text{if } a_t < 0 \\ R_{save} & \text{if } a_t > 0 \end{cases}, \qquad R_{boro} \geq R_{save}.
\end{aligned}
$$

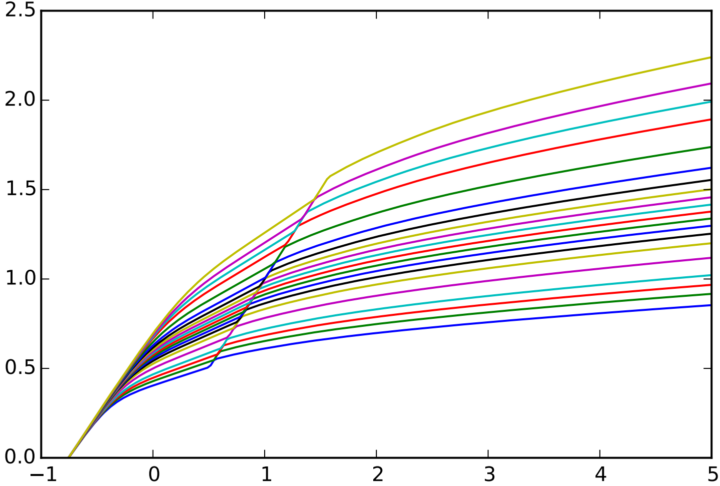
Combination Inheritance: "Kinky Preferences" (3/4)

`ConsKinkyPrefSolver` inherits from two parent classes. Entirety of the code for the solver:

```
class ConsKinkyPrefSolver(ConsPrefShockSolver,ConsKinkedRsolver):
    def __init__(self,solution_next,...):
        ConsKinkedRsolver.__init__(self,solution_next,...)
        self.PrefShkPrbs = PrefShkDstn[0]
        self.PrefShkVals = PrefShkDstn[1]
```

Combination Inheritance: “Kinky Preferences” (4/4)

Back



Macroeconomics in HARK (1/5)

- Some inputs to micro models are exogenous to each agent...
- ...But endogenous to collective whole of agents
- Might be static quantities or dynamic processes

Macroeconomics in HARK (1/5)

- Some inputs to micro models are exogenous to each agent...
- ...But endogenous to collective whole of agents
- Might be static quantities or dynamic processes
- Equilibrium: consistency between what agents *believe* the endogenous objects are and what values / dynamic processes *actually occur* when agents act on those beliefs
- Fixed point in the space of beliefs

Macroeconomics in HARK (1/5)

- Some inputs to micro models are exogenous to each agent...
- ...But endogenous to collective whole of agents
- Might be static quantities or dynamic processes
- Equilibrium: consistency between what agents *believe* the endogenous objects are and what values / dynamic processes *actually occur* when agents act on those beliefs
- Fixed point in the space of beliefs
- Need a representation of beliefs about endogenous objects
- And a rule for how agents form beliefs from observing history

Macroeconomics in HARK (2/5)

"The computational algorithm has two key features. First, it is based on bounded rationality in the sense that we endow agents with boundedly rational perceptions of how the aggregate state evolves... Second, we use solution by simulation, which works as follows: (i) given the boundedly rational perceptions, we solve the individuals' problems using standard dynamic programming methods; (ii) we draw individual and aggregate shocks over time for a large number of individuals; (iii) ...we generate a time series for all aggregates; and finally (iv) we compare the perceptions about the aggregates to those in the actual simulations, and these perceptions are then updated. We think this approach... can be productive for other applications."

–Per Krusell and Tony Smith (2006)

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs
2. Simulate many agents for many periods by looping on:

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs
2. Simulate many agents for many periods by looping on:
   - `sow`: Distribute current aggregate variables to agents

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs
2. Simulate many agents for many periods by looping on:
   - `sow`: Distribute current aggregate variables to agents
   - `cultivate`: Agents act according to their micro solution

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs
2. Simulate many agents for many periods by looping on:
   - `sow`: Distribute current aggregate variables to agents
   - `cultivate`: Agents act according to their micro solution
   - `reap`: Collect some individual variables from the agents

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs
2. Simulate many agents for many periods by looping on:
    - `sow`: Distribute current aggregate variables to agents
    - `cultivate`: Agents act according to their micro solution
    - `reap`: Collect some individual variables from the agents
    - `mill`: Generate aggregate variables from individual vars

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs
2. Simulate many agents for many periods by looping on:
   - `sow`: Distribute current aggregate variables to agents
   - `cultivate`: Agents act according to their micro solution
   - `reap`: Collect some individual variables from the agents
   - `mill`: Generate aggregate variables from individual vars
   - `store`: Record some information in a "history"

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (3/5)

HARK operationalizes K-S method with a farming metaphor:

1. Solve agents' microeconomic problem for some beliefs
2. Simulate many agents for many periods by looping on:
   - `sow`: Distribute current aggregate variables to agents
   - `cultivate`: Agents act according to their micro solution
   - `reap`: Collect some individual variables from the agents
   - `mill`: Generate aggregate variables from individual vars
   - `store`: Record some information in a "history"
3. Use history to update beliefs about endogenous objects

`Market`'s method `solve` loops on this process until convergence.

Macroeconomics in HARK (4/5)

Attributes of a `Market` instance (or subclass):

- `agents`: List of `AgentType` instances in market
- `dyn_vars`: Names of the endogenous objects

Macroeconomics in HARK (4/5)

Attributes of a `Market` instance (or subclass):

- `agents`: List of `AgentType` instances in market
- `dyn_vars`: Names of the endogenous objects
- `sow_vars`: Names of aggregate variables
- `reap_vars`: Individual variables that form aggregates
- `track_vars`: Aggregates that need to be recorded in history

Macroeconomics in HARK (4/5)

Attributes of a `Market` instance (or subclass):

- `agents`: List of `AgentType` instances in market
- `dyn_vars`: Names of the endogenous objects
- `sow_vars`: Names of aggregate variables
- `reap_vars`: Individual variables that form aggregates
- `track_vars`: Aggregates that need to be recorded in history
- `millRule`: Function that transforms ind $\longrightarrow$ agg variables
- `calcDynamics`: Function that transforms history into beliefs

Macroeconomics in HARK (5/5)

Extra methods of a `Market`-compatible `AgentType`:

- `marketAction`: What agents *do* to generate `reap_vars`. Often just simulate one period with `simOnePeriod`

Trivial to add more *ex ante* heterogeneity: just add more `AgentType` instances to `agents`!

Macroeconomics in HARK (5/5)

Extra methods of a `Market`-compatible `AgentType`:

- `marketAction`: What agents *do* to generate `reap_vars`. Often just simulate one period with `simOnePeriod`
- `reset`: How to initialize for a new history: reset states

Trivial to add more *ex ante* heterogeneity: just add more `AgentType` instances to `agents`!

Consumption-Saving with Aggregate Productivity Shocks

$$
\begin{aligned}
v_t(m_t, M_t) &= \max_{c_t} u(c_t) + \beta \cancel{D}_{t+1} \mathbb{E}[v_{t+1}(m_{t+1}, M_{t+1})], \\
a_t &= m_t - c_t, \qquad a_t \geq 0, \\
m_{t+1} &= \frac{R_{t+1} a_t}{\Gamma_{t+1} \psi_{t+1} \Psi_{t+1}} + W_{t+1} \theta_{t+1} \Theta_{t+1} \ell, \\
A_t = \mathbf{A}(M_t), & \qquad k_{t+1} = (1-\delta) A_t / (\Psi_{t+1} \ell), \\
R_{t+1} = \mathbf{R}(k_{t+1}/\Theta_{t+1}), & \qquad W_{t+1} = \mathbf{W}(k_{t+1}/\Theta_{t+1}), \\
M_{t+1} &= R_{t+1} k_{t+1} + W_{t+1} \Theta_{t+1} \ell \\
\theta_{t+1} \sim F_{\theta t+1}, & \qquad \psi_{t+1} \sim F_{\psi t+1}, \quad \mathbb{E}[\psi_{t+1}] = 1, \\
\Theta_{t+1} \sim F_{\Theta}, & \qquad \Psi_{t+1} \sim F_{\Psi}, \quad \mathbb{E}[\Psi_{t+1}] = \mathbb{E}[\Theta_{t+1}] = 1.
\end{aligned}
$$

Consumption-Saving with Aggregate Productivity Shocks

Some totally new inputs for an `AggShockConsumerType`:

- `Rfunc` and `wFunc`: Factor payments as function of k_t

`IncomeDstn` combines idiosyncratic and aggregate shocks:

Consumption-Saving with Aggregate Productivity Shocks

Some totally new inputs for an `AggShockConsumerType`:

- `Rfunc` and `wFunc`: Factor payments as function of k_t
- `Mgrid`: Grid of M_t state values (sort of constructed)

`IncomeDstn` combines idiosyncratic and aggregate shocks:

Consumption-Saving with Aggregate Productivity Shocks

Some totally new inputs for an `AggShockConsumerType`:

- `Rfunc` and `wFunc`: Factor payments as function of k_t
- `Mgrid`: Grid of M_t state values (sort of constructed)
- `Afunc`: $\mathbb{E}[A_t | M_t] = \mathbf{A}(M_t)$

`IncomeDstn` combines idiosyncratic and aggregate shocks:

Consumption-Saving with Aggregate Productivity Shocks

Some totally new inputs for an `AggShockConsumerType`:

- `Rfunc` and `wFunc`: Factor payments as function of k_t
- `Mgrid`: Grid of M_t state values (sort of constructed)
- `Afunc`: $\mathbb{E}[A_t|M_t] = \mathbf{A}(M_t)$

`IncomeDstn` combines idiosyncratic and aggregate shocks:

- Discrete approximation to aggregate shock distribution constructed like idiosyncratic shocks: `PermShkAggStd`, `TranShkAggStd`, `PermShkAggCount`, `TranShkAggCount`

Consumption-Saving with Aggregate Productivity Shocks

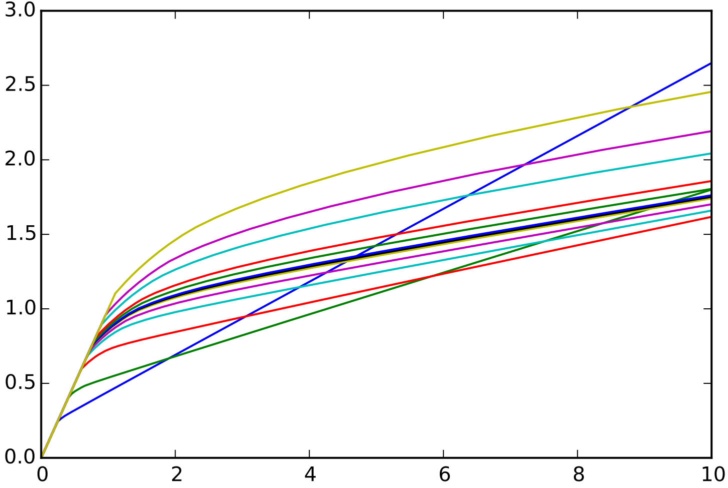
Some totally new inputs for an `AggShockConsumerType`:

- `Rfunc` and `wFunc`: Factor payments as function of k_t
- `Mgrid`: Grid of M_t state values (sort of constructed)
- `Afunc`: $\mathbb{E}[A_t|M_t] = \mathbf{A}(M_t)$

`IncomeDstn` combines idiosyncratic and aggregate shocks:

- Discrete approximation to aggregate shock distribution constructed like idiosyncratic shocks: `PermShkAggStd`, `TranShkAggStd`, `PermShkAggCount`, `TranShkAggCount`
- `IncomeDstn` has five elements: probs, idio shocks, agg shocks

Consumption-Saving with Aggregate Productivity Shocks



Cobb-Douglas Economy in the `Market` Framework

`CobbDouglasEconomy` is a subclass of `Market`:

- `sow`: Distribute $(M_t, R_t, W_t, \Theta_t, \Psi_t)$ to consumers

Loop that process for (say) 1000 periods

Cobb-Douglas Economy in the `Market` Framework

`CobbDouglasEconomy` is a subclass of `Market`:

- `sow`: Distribute $(M_t, R_t, W_t, \Theta_t, \Psi_t)$ to consumers
- `cultivate`: Consumers draw (θ_t, ψ_t), choose c_t

Loop that process for (say) 1000 periods

Cobb-Douglas Economy in the `Market` Framework

`CobbDouglasEconomy` is a subclass of `Market`:

- `sow`: Distribute $(M_t, R_t, W_t, \Theta_t, \Psi_t)$ to consumers
- `cultivate`: Consumers draw (θ_t, ψ_t), choose c_t
- `reap`: Collect assets a_t and productivity P_t from consumers

Loop that process for (say) 1000 periods

Cobb-Douglas Economy in the `Market` Framework

`CobbDouglasEconomy` is a subclass of `Market`:

- `sow`: Distribute $(M_t, R_t, W_t, \Theta_t, \Psi_t)$ to consumers
- `cultivate`: Consumers draw (θ_t, ψ_t), choose c_t
- `reap`: Collect assets a_t and productivity P_t from consumers
- `mill`: Calc A_{t+1}, draw $(\Theta_{t+1}, \Psi_{t+1})$, calc k_{t+1}, M_{t+1}, get (R_{t+1}, W_{t+1})

Loop that process for (say) 1000 periods

Cobb-Douglas Economy in the `Market` Framework

`CobbDouglasEconomy` is a subclass of `Market`:

- `sow`: Distribute $(M_t, R_t, W_t, \Theta_t, \Psi_t)$ to consumers
- `cultivate`: Consumers draw (θ_t, ψ_t), choose c_t
- `reap`: Collect assets a_t and productivity P_t from consumers
- `mill`: Calc A_{t+1}, draw $(\Theta_{t+1}, \Psi_{t+1})$, calc k_{t+1}, M_{t+1}, get (R_{t+1}, W_{t+1})
- `store`: Record M_t and A_t in their histories

Loop that process for (say) 1000 periods

Cobb-Douglas Economy in the `Market` Framework

`CobbDouglasEconomy` is a subclass of `Market`:

- `sow`: Distribute $(M_t, R_t, W_t, \Theta_t, \Psi_t)$ to consumers
- `cultivate`: Consumers draw (θ_t, ψ_t), choose c_t
- `reap`: Collect assets a_t and productivity P_t from consumers
- `mill`: Calc A_{t+1}, draw $(\Theta_{t+1}, \Psi_{t+1})$, calc k_{t+1}, M_{t+1}, get (R_{t+1}, W_{t+1})
- `store`: Record M_t and A_t in their histories

Loop that process for (say) 1000 periods

- `calcDynamics`: Regress $log(A_t)$ on $log(M_t)$, make new $\mathbf{A}$

Cobb-Douglas Economy in the `Market` Framework

`CobbDouglasEconomy` is a subclass of `Market`:

- `sow`: Distribute $(M_t, R_t, W_t, \Theta_t, \Psi_t)$ to consumers
- `cultivate`: Consumers draw (θ_t, ψ_t), choose c_t
- `reap`: Collect assets a_t and productivity P_t from consumers
- `mill`: Calc A_{t+1}, draw $(\Theta_{t+1}, \Psi_{t+1})$, calc k_{t+1}, M_{t+1}, get (R_{t+1}, W_{t+1})
- `store`: Record M_t and A_t in their histories

Loop that process for (say) 1000 periods

- `calcDynamics`: Regress $log(A_t)$ on $log(M_t)$, make new $\mathbf{A}$
- Distribute new $\mathbf{A}$ to consumers as `Afunc`

Other Applications of `Market`

Krusell and Smith were right: method is applicable to other topics

- Premiums of medical insurance contracts: actuarial constraint maps *who* buys each contract to break even premium, subject to informational constraints (sex, age, health, etc)

Other Applications of `Market`

Krusell and Smith were right: method is applicable to other topics

- Premiums of medical insurance contracts: actuarial constraint maps *who* buys each contract to break even premium, subject to informational constraints (sex, age, health, etc)
- Papageorge et al risky sex framework: probability of contracting HIV from risky sex act depends on HIV infection rate and risky sex choices of the population

Back

Other Applications of `Market`

Krusell and Smith were right: method is applicable to other topics

- Premiums of medical insurance contracts: actuarial constraint maps *who* buys each contract to break even premium, subject to informational constraints (sex, age, health, etc)
- Papageorge et al risky sex framework: probability of contracting HIV from risky sex act depends on HIV infection rate and risky sex choices of the population
- Agent-to-agent interaction: could `sow` a permutation of what is `reaped`: imperfect knowledge, contagion of information, moves closer to "agent-based modeling"

Back

References I

Carroll, Christopher D., Edmund Crawley, Jiri Slacalek, Kiichi Tokuoka, and Matthew N. White (2018): "Sticky Expectations and Consumption Dynamics," *Manuscript, Johns Hopkins University*.

Carroll, Christopher D., Jiri Slacalek, Kiichi Tokuoka, and Matthew N. White (2017): "The Distribution of Wealth and the Marginal Propensity to Consume," *Quantitative Economics*, 8, 977–1020, At `http://econ.jhu.edu/people/ccarroll/papers/cstwMPC`.

Fagereng, Andreas, Martin B. Holm, and Gisle J. Natvik (2017): "MPC Heterogeneity and Household Balance Sheets," discussion paper, Statistics Norway.