Endogenous grid method

Marcin Lewandowski

January 14, 2025

What is the endogenous grid method?

- A highly efficient method to solve a wide range of economics models
- Developed by Chris Carroll in 2006
- It exploits the FOC, but does not require solving numerically the nonlinear equation
- In this class we use it to solve simplest heterogenous agent models:
    - Heterogeneity comes from the productivity endowment
    - We focus on the agent consumption-saving problem

The simplest consumption-saving problem:

Let's (again) think of a very simple problem.

- Agents live for 2 periods (young and old)

The simplest consumption-saving problem:

Let's (again) think of a very simple problem.

- Agents live for 2 periods (young and old)
- Income is a stochastic variable.

The simplest consumption-saving problem:

Let's (again) think of a very simple problem.

- Agents live for 2 periods (young and old)
- Income is a stochastic variable.
- Agents (can) enter their life with some initial level of assets $a \neq 0$.

The simplest consumption-saving problem:

Let's (again) think of a very simple problem.

- Agents live for 2 periods (young and old)
- Income is a stochastic variable.
- Agents (can) enter their life with some initial level of assets $a \neq 0$.
- And cannot die with debt $a'' \geq 0$!

The simplest consumption-saving problem:

Let's (again) think of a very simple problem.

- Agents live for 2 periods (young and old)
- Income is a stochastic variable.
- Agents (can) enter their life with some initial level of assets $a \neq 0$.
- And cannot die with debt $a'' \geq 0$!

The simplest consumption-saving problem:

Let's (again) think of a very simple problem.

- Agents live for 2 periods (young and old)
- Income is a stochastic variable.
- Agents (can) enter their life with some initial level of assets $a \neq 0$.
- And cannot die with debt $a'' \geq 0$!

Notation, let:

- c: young consumption; c': old consumption
- a: young assets; a': old assets; (Note optimal $a'' = 0$).
- y: young income realization; y': old income realization.
- $R = 1 + r$, where r is the interest rate
- β: discount factor

Solving the model

Our goal is to solve a model i.e. find a policy *functions*:

- $c(a, y)$: young optimal consumption for particular level of a, y
- $c'(a', y')$: old optimal consumption for particular level of a, y
- $a'(a, y)$: young optimal level of assets saved for particular level of a, y
- $a''(a', y')$: old optimal level of assets saved for particular level of a, y

Note: In the context of 2-period model $a''(a, y)$, $c'(a', y')$ are reeeeeally obvious...

Solving the model

Our goal is to solve a model i.e. find a policy *functions*:

- $c(a, y)$: young optimal consumption for particular level of a, y
- $c'(a', y')$: old optimal consumption for particular level of a, y
- $a'(a, y)$: young optimal level of assets saved for particular level of a, y
- $a''(a', y')$: old optimal level of assets saved for particular level of a, y

Note: In the context of 2-period model $a''(a, y)$, $c'(a', y')$ are reeeeeally obvious...

We are really after $c(a, y)$ and $a'(a, y)$, EGM is one possible way to approach this problem.

EGM intro: Let's use the dynamic programming

Instead of solving the problem through lifetime utility maximization, we can use dynamic programming.
Define the old value function as:

$$V_2(a', y') = \max_{a''} u\left(Ra' + y' - a''\right)$$
$$V_2(a', y') = u\left(Ra' + y'\right) = u\left(c'\right)$$

- Note 1: For old-age decision-maker a', y' are given!

EGM intro: Let's use the dynamic programming

Instead of solving the problem through lifetime utility maximization, we can use dynamic programming.

Define the old value function as:

$$V_2(a', y') = \max_{a''} u\left(Ra' + y' - a''\right)$$
$$V_2(a', y') = u\left(Ra' + y'\right) = u\left(c'\right)$$

- Note 1: For old-age decision-maker a', y' are given!
- Note 2: $c'(a', y') = Ra' + y'$

EGM intro: Let's use the dynamic programming

Instead of solving the problem through lifetime utility maximization, we can use dynamic programming.
Define the old value function as:

$$V_2(a', y') = \max_{a''} u\left(Ra' + y' - a''\right)$$
$$V_2(a', y') = u\left(Ra' + y'\right) = u\left(c'\right)$$

- Note 1: For old-age decision-maker a', y' are given!
- Note 2: $c'(a', y') = Ra' + y'$
- Note 3: $a''(a', y') = 0$

EGM intro: Let's use the dynamic programming

Instead of solving the problem through lifetime utility maximization, we can use dynamic programming.

Define the old value function as:

$$V_2(a', y') = \max_{a''} u\left(Ra' + y' - a''\right)$$

$$V_2(a', y') = u\left(Ra' + y'\right) = u\left(c'\right)$$

- Note 1: For old-age decision-maker a', y' are given!
- Note 2: $c'(a', y') = Ra' + y'$
- Note 3: $a''(a', y') = 0$

EGM intro: Let's use the dynamic programming

Instead of solving the problem through lifetime utility maximization, we can use dynamic programming.

Define the old value function as:

$$V_2(a', y') = \max_{a''} u\left(Ra' + y' - a''\right)$$

$$V_2(a', y') = u\left(Ra' + y'\right) = u\left(c'\right)$$

- Note 1: For old-age decision-maker a', y' are given!
- Note 2: $c'(a', y') = Ra' + y'$
- Note 3: $a''(a', y') = 0$

Define the young value function as:

$$V_1(a, y) \equiv \max_{a'} \left\{ u\left(Ra + y - a'\right) + \beta \mathbb{E}_{y'|y}\left[V_2(a', y')\right] \right\}$$

EGM intro: Let's use the dynamic programming

$$V_1(a, y) \equiv \max_{a'} \left\{ u\left(Ra + y - a'\right) + \beta \mathbb{E}_{y'|y}\left[V_2(a', y')\right] \right\}$$

FOC:

$$\begin{aligned} u'(c) &= \beta \frac{\partial}{\partial a'} \mathbb{E}_{y'|y}\left[V_2(a', y')\right] \\ &= \beta \mathbb{E}_{y'|y}\left[\frac{\partial}{\partial a'} V_2(a', y')\right] \end{aligned} \tag{1}$$

EGM intro: Let's use the dynamic programming

$$V_1(a, y) \equiv \max_{a'} \left\{ u\left(Ra + y - a'\right) + \beta \mathbb{E}_{y'|y}\left[V_2(a', y')\right] \right\}$$

FOC:

$$\begin{aligned} u'(c) &= \beta \frac{\partial}{\partial a'} \mathbb{E}_{y'|y}\left[V_2(a', y')\right] \\ &= \beta \mathbb{E}_{y'|y}\left[\frac{\partial}{\partial a'} V_2(a', y')\right] \end{aligned} \tag{1}$$

Note that we can find $\frac{\partial}{\partial a'} V_2(a', y')$ easily:

$$\frac{\partial}{\partial a'} V_2(a', y') = Ru'\left(Ra' + y'\right) = Ru'\left(c'\right) \tag{2}$$

EGM intro: Let's use the dynamic programming

$$V_1(a,y) \equiv \max_{a'} \left\{ u\left(Ra + y - a'\right) + \beta \mathbb{E}_{y'|y}\left[V_2(a',y')\right] \right\}$$

FOC:

$$\begin{aligned} u'(c) &= \beta \frac{\partial}{\partial a'} \mathbb{E}_{y'|y}\left[V_2(a',y')\right] \\ &= \beta \mathbb{E}_{y'|y}\left[\frac{\partial}{\partial a'} V_2(a',y')\right] \end{aligned} \tag{1}$$

Note that we can find $\frac{\partial}{\partial a'} V_2(a',y')$ easily:

$$\frac{\partial}{\partial a'} V_2(a',y') = Ru'\left(Ra' + y'\right) = Ru'\left(c'\right) \tag{2}$$

Inserting (2) into (1) we get:

$$u'(c) = \beta \mathbb{E}_{y'|y}\left[Ru'\left(c'\right)\right]$$

Quick task in Julia

To get the sense of the objects we are dealing with implement:

1. Old-age policy functions:
   - $c'(a', y') = Ra' + y'$
   - $a''(a', y') = 0$
2. The first derivative of the value function:
   - $\frac{\partial}{\partial a'} V_2(a', y') = Ru'\left(Ra' + y'\right) = Ru'\left(c'\right)$

into Julia

Recall our main goal:

Find policy *functions*:

- $c(a, y)$: young optimal consumption for particular a, y
- $a'(a, y)$: young optimal level of assets saved for particular a, y

Recall our main goal:

Find policy *functions*:

- $c(a, y)$: young optimal consumption for particular a, y
- $a'(a, y)$: young optimal level of assets saved for particular a, y

A naïve way to do it:

- Construct a grid over a and y

Recall our main goal:

Find policy *functions*:

- $c(a, y)$: young optimal consumption for particular a, y
- $a'(a, y)$: young optimal level of assets saved for particular a, y

A naïve way to do it:

- Construct a grid over a and y
- Iterate over all possible current level of assets and income (a, y)

Recall our main goal:

Find policy *functions*:

- $c(a, y)$: young optimal consumption for particular a, y
- $a'(a, y)$: young optimal level of assets saved for particular a, y

A naïve way to do it:

- Construct a grid over a and y
- Iterate over all possible current level of assets and income (a, y)
- For each pair (a, y) iterate over all possible levels of future assets and income (a', y')

Recall our main goal:

Find policy *functions*:

- $c(a, y)$: young optimal consumption for particular a, y
- $a'(a, y)$: young optimal level of assets saved for particular a, y

A naïve way to do it:

- Construct a grid over a and y
- Iterate over all possible current level of assets and income (a, y)
- For each pair (a, y) iterate over all possible levels of future assets and income (a', y')
- Maximum value of $u\left(Ra + y - a'\right) + \beta \mathbb{E}_{y'|y}\left[V_2(a', y')\right]$ implies we found the best a' given a, y.

Endogenous grid method:

EGM inverts this logic and asks:

Suppose you find it optimal to save a'. What level of c, a would justify such a choice?

Endogenous grid method:

EGM inverts this logic and asks:

Suppose you find it optimal to save a'. What level of c, a would justify such a choice?

Remember the optimal a' has to obey the following equation!

$$u'(c) = \beta \mathbb{E}_{y'|y}\left[Ru'\left(c'\right)\right]$$

(provided the agent is not liquidity constrained)

CRRA utility function

For simplicity let's assume CRRA utility function $\sigma \neq 1$:

$$u(c) = \frac{c^{1-\sigma} - 1}{1 - \sigma}$$
$$u'(c) = c^{-\sigma}$$
$$u'^{-1}(c) = c^{-\frac{1}{\sigma}}$$

EGM logic:

- Step 1: Assume some optimal a', and current y which implies:

$$c^{-\sigma} = \beta R \mathbb{E}_{y'|y}\left[\left(c'\left(a', y'\right)\right)^{-\sigma}\right]$$

EGM logic:

- Step 1: Assume some optimal a', and current y which implies:

$$c^{-\sigma} = \beta R \mathbb{E}_{y'|y}\left[\left(c'\left(a', y'\right)\right)^{-\sigma}\right]$$

- Step 2: Invert it to find c which would justify such a':

$$\begin{aligned} (c)^{-\sigma} &= \beta R \mathbb{E}_{y'|y}\left[c'^{-\sigma}\right] \\ c &= \left[\beta R \mathbb{E}_{y'|y}\left[c'^{-\sigma}\right]\right]^{\frac{1}{-\sigma}} \end{aligned}$$

EGM logic:

- Step 1: Assume some optimal a', and current y which implies:

$$c^{-\sigma} = \beta R \mathbb{E}_{y'|y}\left[\left(c'\left(a',y'\right)\right)^{-\sigma}\right]$$

- Step 2: Invert it to find c which would justify such a':

$$\begin{aligned}(c)^{-\sigma} &= \beta R \mathbb{E}_{y'|y}\left[c'^{-\sigma}\right] \\ c &= \left[\beta R \mathbb{E}_{y'|y}\left[c'^{-\sigma}\right]\right]^{\frac{1}{-\sigma}}\end{aligned}$$

- Step 3: Knowing optimal a' and c as well as (current) y, find a:

$$\begin{aligned}Ra + y - a' &= c \\ a &= \frac{c - y + a'}{R}\end{aligned}$$

EGM algorithm:

1. Start for a grid of possible a'

EGM algorithm:

1. Start for a grid of possible a'
2. For each a' use FOC to find c which would justify such a'

EGM algorithm:

1. Start for a grid of possible a'
2. For each a' use FOC to find c which would justify such a'
3. Use the budget constraint to find a

EGM algorithm:

1. Start for a grid of possible a'
2. For each a' use FOC to find c which would justify such a'
3. Use the budget constraint to find a
4. We now have endogenous grid for a. We have $a(a', y)$, where a', y are on the exogenous grid.

EGM algorithm:

1. Start for a grid of possible a'
2. For each a' use FOC to find c which would justify such a'
3. Use the budget constraint to find a
4. We now have endogenous grid for a. We have $a(a', y)$, where a', y are on the exogenous grid.
5. We need to interpolate to get $a'(a, y)$, where a is on the exogenous grid.

EGM algorithm:

1. Start for a grid of possible a'
2. For each a' use FOC to find c which would justify such a'
3. Use the budget constraint to find a
4. We now have endogenous grid for a. We have $a(a', y)$, where a', y are on the exogenous grid.
5. We need to interpolate to get $a'(a, y)$, where a is on the exogenous grid.
6. Now we have $a'(a, y)$, $c(a, y)$ is easy to obtain.

EGM algorithm: Liquidity constrained agents

1. For those agents:

$$u'(c) > \beta \mathbb{E}_{y'|y} \left[R u' \left(c' \left(a', y' \right) \right) \right]$$

EGM algorithm: Liquidity constrained agents

1. For those agents:

$$u'(c) > \beta \mathbb{E}_{y'|y}\left[Ru'\left(c'\left(a', y'\right)\right)\right]$$

2. Define a_grid as our grid. Here a_grid [1] is the lowest possible level.

EGM algorithm: Liquidity constrained agents

1. For those agents:

$$u'(c) > \beta \mathbb{E}_{y'|y} \left[R u' \left(c' \left(a', y' \right) \right) \right]$$

2. Define a_grid as our grid. Here a_grid [1] is the lowest possible level.
3. Recall: $a(a', y)$ is a function which recovers a from the level of a' chosen.

EGM algorithm: Liquidity constrained agents

1. For those agents:

$$u'(c) > \beta \mathbb{E}_{y'|y} \left[R u' \left(c' \left(a', y' \right) \right) \right]$$

2. Define a_grid as our grid. Here a_grid [1] is the lowest possible level.
3. Recall: $a(a', y)$ is a function which recovers a from the level of a' chosen.
4. Agent is liquidity constrained for all $a \in \text{a\_grid} : a < a(\text{a\_grid}[1], y)$

EGM algorithm: Liquidity constrained agents

1. For those agents:

$$u'(c) > \beta \mathbb{E}_{y'|y}\left[Ru'\left(c'\left(a', y'\right)\right)\right]$$

2. Define a_grid as our grid. Here a_grid [1] is the lowest possible level.
3. Recall: $a(a', y)$ is a function which recovers a from the level of a' chosen.
4. Agent is liquidity constrained for all $a \in \text{a\_grid} : a < a(\text{a\_grid}[1], y)$
5. For such an agent:

EGM algorithm: Liquidity constrained agents

1. For those agents:

$$u'(c) > \beta \mathbb{E}_{y'|y} \left[R u' \left(c' \left(a', y' \right) \right) \right]$$

2. Define a_grid as our grid. Here a_grid [1] is the lowest possible level.
3. Recall: $a(a', y)$ is a function which recovers a from the level of a' chosen.
4. Agent is liquidity constrained for all $a \in \text{a\_grid} : a < a(\text{a\_grid}[1], y)$
5. For such an agent:
   - $a'(a, y) = 0$

EGM algorithm: Liquidity constrained agents

1. For those agents:

$$u'(c) > \beta \mathbb{E}_{y'|y}\left[Ru'\left(c'\left(a', y'\right)\right)\right]$$

2. Define a_grid as our grid. Here a_grid [1] is the lowest possible level.
3. Recall: $a(a', y)$ is a function which recovers a from the level of a' chosen.
4. Agent is liquidity constrained for all $a \in \text{a\_grid} : a < a(\text{a\_grid}\left[1\right], y)$
5. For such an agent:
    - $a'(a, y) = 0$
    - $c(a, y) = Ra + y$