# Dynamic Programming Exercise Class 8

Jacob Røpke

march 2025

# Return of the Deaton model

We focus on how we can efficiently solve consumption-savings models like the one below. We focus on finite horizon for now:

$$V_t(M_t) = \max_{C_t} \frac{C_t^{1-\rho}}{1-\rho} + \beta E_t[V_{t+1}(M_t + 1)]$$

s.t

$$M_{t+1} = R(M_t - C_t) + Y_{t+1}$$

$$Y_{t+1} = \exp(\xi_{t+1})$$

$$\xi_{t+1} \sim N(\mu, \sigma_\xi^2)$$

$$A_t = M_t - C_t \geq 0$$

**FOC:**

$$C_t^{-\rho} = \beta R E_t[C_{t+1}^{-\rho}]$$

# Previously: Value function iteration

Algorithm in finite horizon works as follows:

1. Initialise solution grid for $M_t$.

# Previously: Value function iteration

Algorithm in finite horizon works as follows:

1. Initialise solution grid for $M_t$.
2. Last period of life: Consume everything and approximate terminal value function by just the utility

# Previously: Value function iteration

Algorithm in finite horizon works as follows:

1. Initialise solution grid for $M_t$.
2. Last period of life: Consume everything and approximate terminal value function by just the utility
3. Start backward induction from period $T-2$:

# Previously: Value function iteration

Algorithm in finite horizon works as follows:

1. Initialise solution grid for $M_t$.
2. Last period of life: Consume everything and approximate terminal value function by just the utility
3. Start backward induction from period $T-2$:
   3.1 Solve Right-hand-side (RHS) of bellman equation to get optimal consumption $C_t^*$ while using value-function approx. at $t+1$:

$$\max_{C_t} \frac{C_t^{1-\rho}}{1-\rho} + \beta E_t[V_{t+1}(M_t+1)]$$

# Previously: Value function iteration

Algorithm in finite horizon works as follows:

1. Initialise solution grid for $M_t$.
2. Last period of life: Consume everything and approximate terminal value function by just the utility
3. Start backward induction from period $T-2$:
   3.1 Solve Right-hand-side (RHS) of bellman equation to get optimal consumption $C_t^*$ while using value-function approx. at $t+1$:

   $$\max_{C_t} \frac{C_t^{1-\rho}}{1-\rho} + \beta E_t[V_{t+1}(M_t+1)]$$

   3.2 Save $C_t^*$ into policy.
   3.3 Compute value function as the value of the RHS of the bellman equation for $C_t^*$.

# Time iterations

Similar in concept but instead iterate on euler equation. Algorithm works as follows:

1. Initialise solution grid for $M_t$

# Time iterations

Similar in concept but instead iterate on euler equation. Algorithm works as follows:

1. Initialise solution grid for $M_t$
2. Consume everything in last period.

# Time iterations

Similar in concept but instead iterate on euler equation. Algorithm works as follows:

1. Initialise solution grid for $M_t$
2. Consume everything in last period.
3. Start backward induction from $T - 2$:

# Time iterations

Similar in concept but instead iterate on euler equation. Algorithm works as follows:

1. Initialise solution grid for $M_t$
2. Consume everything in last period.
3. Start backward induction from $T - 2$:
   3.1 Solve the euler equation for optimal $C_t$: $C_t^*$ while using solution for $C_{t+1}(M_t)$ to evaluate the RHS:

$$C_t^{-\rho} = \beta R E_t[C_{t+1}(M_{t+1})^{-\rho}]$$

# Time iterations

Similar in concept but instead iterate on euler equation. Algorithm works as follows:

1. Initialise solution grid for $M_t$
2. Consume everything in last period.
3. Start backward induction from $T - 2$:
    3.1 Solve the euler equation for optimal $C_t$: $C_t^*$ while using solution for $C_{t+1}(M_t)$ to evaluate the RHS:

    $$C_t^{-\rho} = \beta R E_t[C_{t+1}(M_{t+1})^{-\rho}]$$

    3.2 Enforce constraint: If $C_t^* > M_t$ set $C_t^* = M_t$

# Time iterations

Similar in concept but instead iterate on euler equation. Algorithm works as follows:

1. Initialise solution grid for $M_t$
2. Consume everything in last period.
3. Start backward induction from $T-2$:
   3.1 Solve the euler equation for optimal $C_t$: $C_t^*$ while using solution for $C_{t+1}(M_t)$ to evaluate the RHS:

   $$C_t^{-\rho} = \beta R E_t[C_{t+1}(M_{t+1})^{-\rho}]$$

   3.2 Enforce constraint: If $C_t^* > M_t$ set $C_t^* = M_t$
   3.3 Save $C_t^*$ in policy function.

# Endogenous Gridpoint Method (EGM)

Similar to time iterations but we solve euler equation differently:

1. Initialise solution grid for post-decision wealth $A_t = M_t - C_t$

# Endogenous Gridpoint Method (EGM)

Similar to time iterations but we solve euler equation differently:

1. Initialise solution grid for post-decision wealth $A_t = M_t - C_t$
2. Consume everything in last period.

# Endogenous Gridpoint Method (EGM)

Similar to time iterations but we solve euler equation differently:

1. Initialise solution grid for post-decision wealth $A_t = M_t - C_t$
2. Consume everything in last period.
3. Start backward induction from $T-2$:

# Endogenous Gridpoint Method (EGM)

Similar to time iterations but we solve euler equation differently:

1. Initialise solution grid for post-decision wealth $A_t = M_t - C_t$
2. Consume everything in last period.
3. Start backward induction from $T-2$:
   3.1 Compute optimal $C_t$: $C_t^*$ while using grid for $A_t$ to evaluate the RHS of the **inverted Euler equation**:

$$C_t = \left(\beta R E_t[C_{t+1}(M_{t+1})^{-\rho}]\right)^{-\frac{1}{\rho}}$$

# Endogenous Gridpoint Method (EGM)

Similar to time iterations but we solve euler equation differently:

1. Initialise solution grid for post-decision wealth $A_t = M_t - C_t$
2. Consume everything in last period.
3. Start backward induction from $T-2$:
    3.1 Compute optimal $C_t$: $C_t^*$ while using grid for $A_t$ to evaluate the RHS of the **inverted Euler equation**:

$$C_t = \left(\beta R E_t[C_{t+1}(M_{t+1})^{-\rho}]\right)^{-\frac{1}{\rho}}$$

    3.2 Compute endogenous cash-on-hand: $M_t^* = C_t^* + A_t$

# Endogenous Gridpoint Method (EGM)

Similar to time iterations but we solve euler equation differently:

1. Initialise solution grid for post-decision wealth $A_t = M_t - C_t$
2. Consume everything in last period.
3. Start backward induction from $T - 2$:
   3.1 Compute optimal $C_t$: $C_t^*$ while using grid for $A_t$ to evaluate the RHS of the **inverted Euler equation**:

$$C_t = \left(\beta R E_t[C_{t+1}(M_{t+1})^{-\rho}]\right)^{-\frac{1}{\rho}}$$

   3.2 Compute endogenous cash-on-hand: $M_t^* = C_t^* + A_t$
   3.3 Save $C_t^*$ in policy function and also save $M_t^*$

# Endogenous Gridpoint Method (EGM)

Similar to time iterations but we solve euler equation differently:

1. Initialise solution grid for post-decision wealth $A_t = M_t - C_t$
2. Consume everything in last period.
3. Start backward induction from $T-2$:
   3.1 Compute optimal $C_t$: $C_t^*$ while using grid for $A_t$ to evaluate the RHS of the **inverted Euler equation**:

$$C_t = \left(\beta R E_t[C_{t+1}(M_{t+1})^{-\rho}]\right)^{-\frac{1}{\rho}}$$

   3.2 Compute endogenous cash-on-hand: $M_t^* = C_t^* + A_t$
   3.3 Save $C_t^*$ in policy function and also save $M_t^*$
4. Add a point at $M_t = 0, C_t^* = 0$ to both the endogenous $M_t$-grid and to the policy function to handle the constraint.