Lecture 1
Second-year Ph.D. course in Household Finance

Roine Vestman[1]

October 24, 2023

[1]Stockholm University, SHoF, and CEPR

What is Household Finance?

- Topics on households' economic behavior, including implications (for macro, financial stability, consumer protection, policy design)
- Diverse in terms methods – borders to Macroeconomics, Labor Economics, Behavioral Economics, and Asset Pricing. Uses broad range of empirical and theoretical methods
- Typically some empirical component of papers (e.g. registry-based data), but pay-off to structural modelling using e.g. dynamic programming is big
- Nice seminar series and conferences:
  - `https://micro-macro-household-finance.co.uk/`
  - `https://cepr.org/research/research-policy-networks/household-finance`
  - `https://www.nber.org/conferences/si-2022-household-finance`

Outline of day 1

- Solving the sequential formulation of a simple consumption-savings problem
- Dynamic programming formulation of a simple yet "realistic" consumer's problem (Carroll, QJE 1997)
  - No-borrowing constraint / tight borrowing constraint
  - Uninsurable income risk (Carroll and Samwick, JME 1997)
- General tips and advice for programming
  - Pros and cons on programming languages
  - The coding process
  - Books, resources, etc
- A quasi-algorithm for solving and simulating the consumer's problem using the endogenous grid point method (Carroll, EL 2006)
- Discussion of the problem set

Outline of day 2

- Extensions to the consumer's problem
  - Epstein-Zin preferences
  - Portfolio choice
  - Pension systems
  - Housing and mortgages
- The curse of dimensionality
- EGM and discrete control variables
- Assistance with problem set

Review: a very basic consumption-savings problem

$$U = \max_{\{C_t, A_t\}_{t=1}^{T}} \sum_{t=1}^{T} \beta^{t-1} u(C_t) \tag{1}$$

subject to

$$C_1 + A_1 = Y_1 + \hat{A} \tag{2}$$

$$C_t + A_t = Y_t + A_{t-1} R \qquad t = 2, 3, ..., T \tag{3}$$

$$A_T = 0 \tag{4}$$

(Notation: $R = 1 + r$.)
Assume $Y_t > 0$ and/or $\hat{A} > 0$.
Notice: no restriction on $\{A_t\}_{t=1}^{T-1}$.

Consolidated budget constraint and first-order condition from Lagrangian formulation

(3) implies:

$$A_{t-1} = (C_t + A_t - Y_t)R^{-1} \tag{5}$$

Substitute forward iteratively into (3) and use $A_T = 0$:

$$\sum_{t=1}^{T} C_t R^{1-t} = \hat{A} + \sum_{t=1}^{T} Y_t R^{1-t} \tag{6}$$

Optimization: Lagrangian and FOC w.r.t C_t and A_t:

$$u'(C_t) = \beta R u'(C_{t+1}) \quad t = 1, 2, \ldots T-1 \tag{7}$$

CRRA:

$$C_t^{-\gamma} = \beta R C_{t+1}^{-\gamma} \quad t = 1, 2, \ldots T-1 \tag{8}$$

$$C_t = (\beta R)^{-1/\gamma} C_{t+1} \quad t = 1, 2, \ldots T-1 \tag{9}$$

Iterate backwards to express C_t as a function of C_1:

$$C_2 = C_1(\beta R)^{1/\gamma} \tag{10}$$

$$C_3 = C_2(\beta R)^{1/\gamma} = C_1(\beta R)^{2/\gamma} \tag{11}$$

$$C_t = C_1(\beta R)^{(t-1)/\gamma} \tag{12}$$

Solving for paths

Substitute (12) into (6):

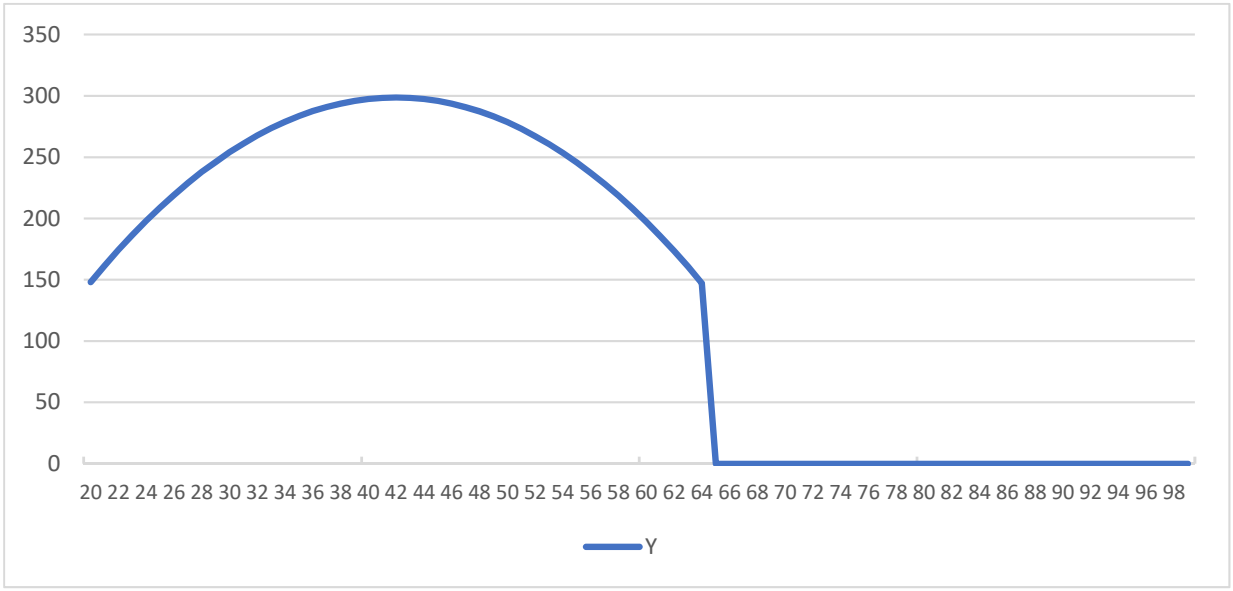
$$C_1 \sum_{t=1}^{T} (\beta R)^{(t-1)/\gamma} R^{1-t} = \hat{A} + \sum_{t=1}^{T} Y_t R^{1-t} \tag{13}$$

This pins down C_1 and hence $\{C_t\}_{t=1}^{T-1}$ from (12). Back out $\{A_t\}_{t=1}^{T-1}$ from (2) and (3).
If $\beta R = 1$ and $Y_t = Y$ then:

$$C = Y + \left(\frac{r}{R}\right)\left[1 - R^{-T}\right]^{-1} \hat{A} \tag{14}$$

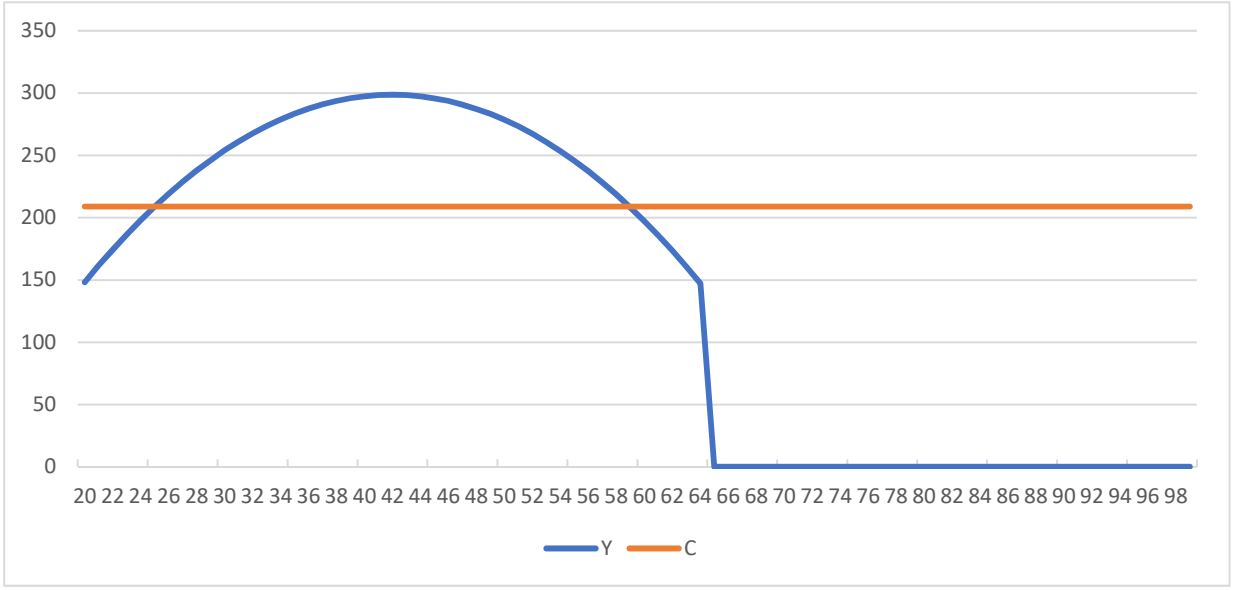
Compare to the expression for $c^{owner} - c^{renter}$ in Sodini, Van Nieuwerburgh, Vestman and von Lilienfeld (AER, 2023).

Log utility and a somewhat realistic income path

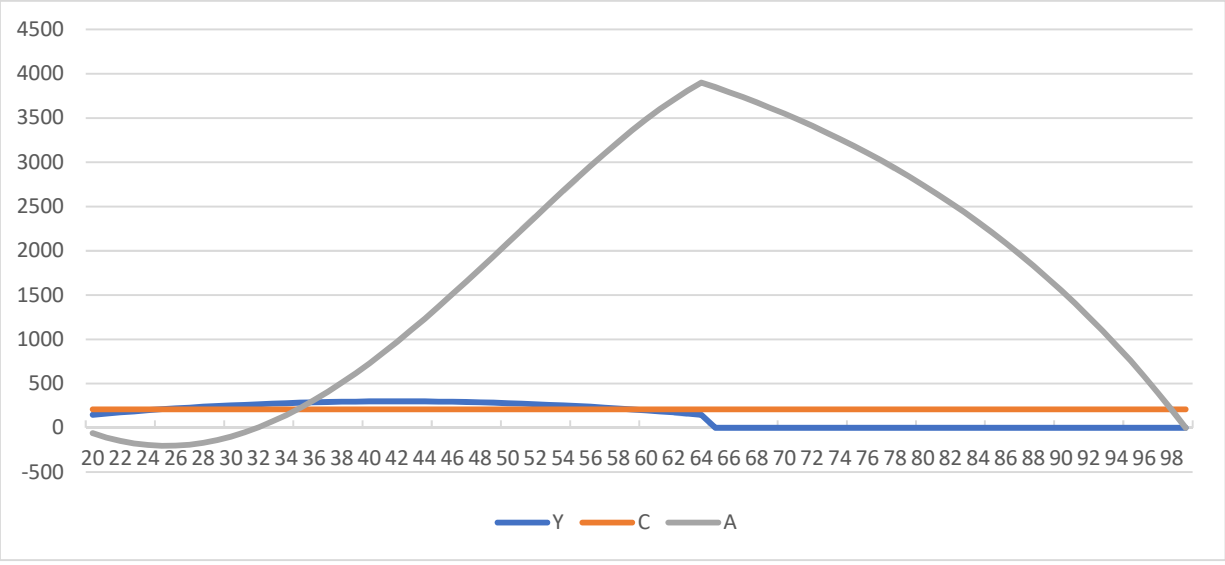


- $t = 1, 2, \ldots, 80$ (age 20–99)
- Retirement at $t = 45$ (age 65)
- Hump-shaped labor income until retirement (unit: SEK 1,000)
- $R = 1.04$, $\beta = R^{-1}$
- $\hat{A} = 0$

Optimal consumption

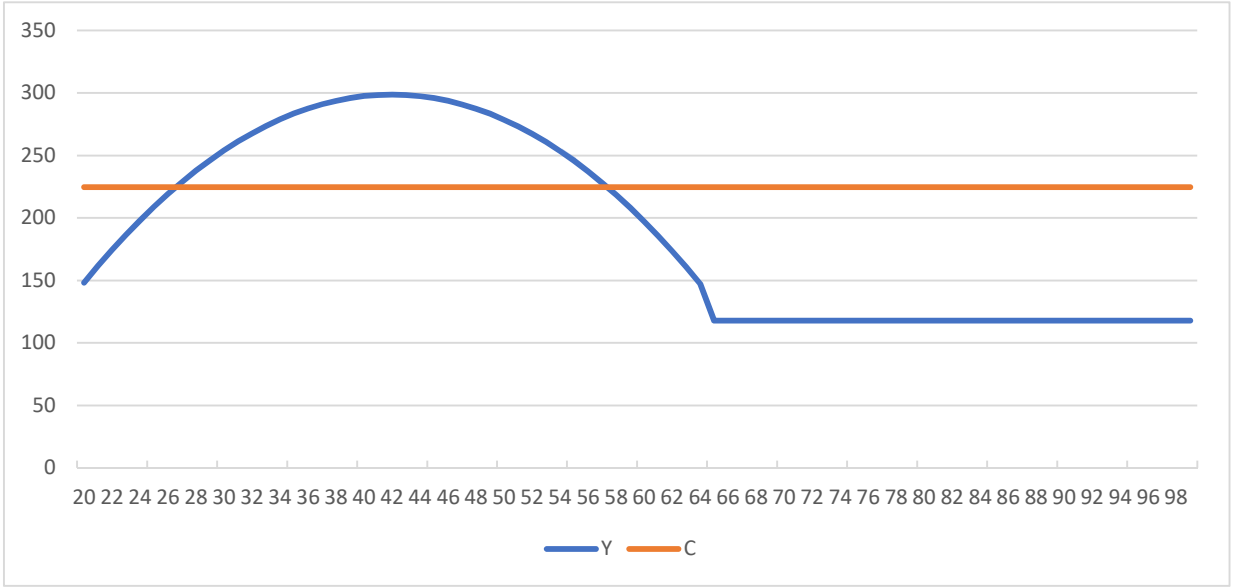


Asset postions are extreme

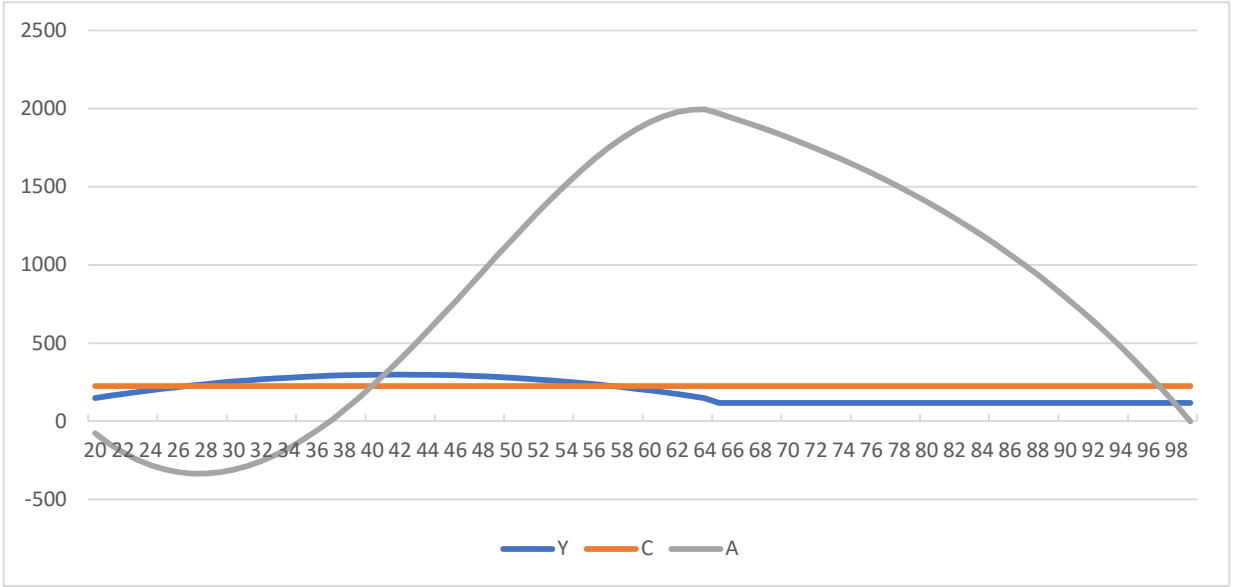


The consumer borrows SEK 203,000 before saving up to 3.9 million.

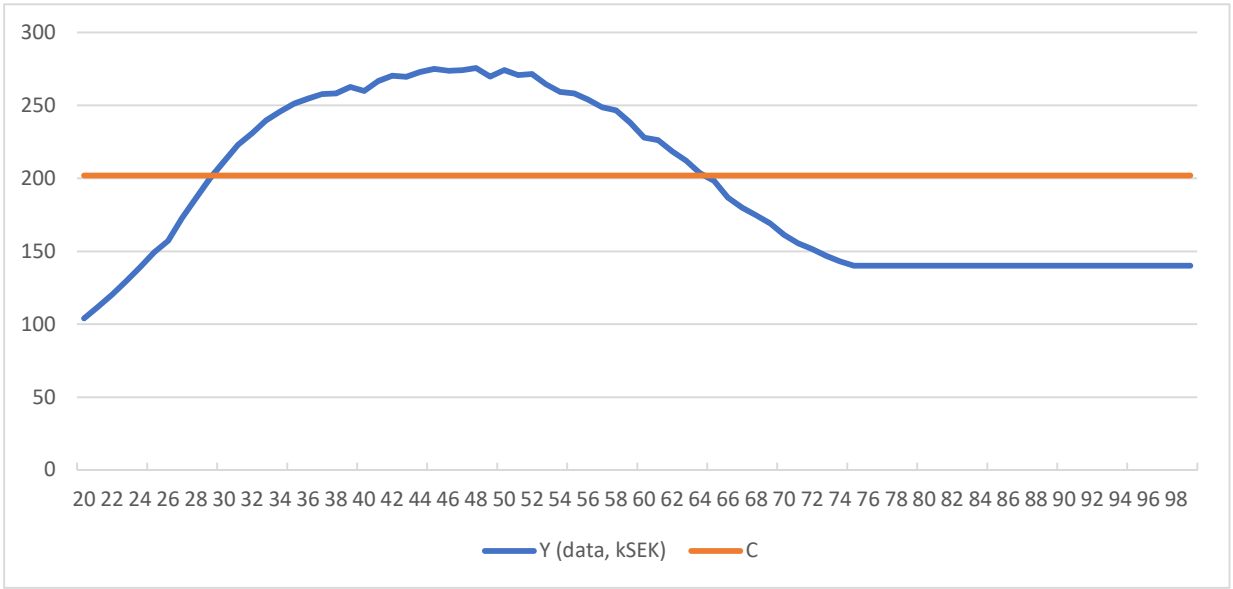
80% replacement rate in retirement



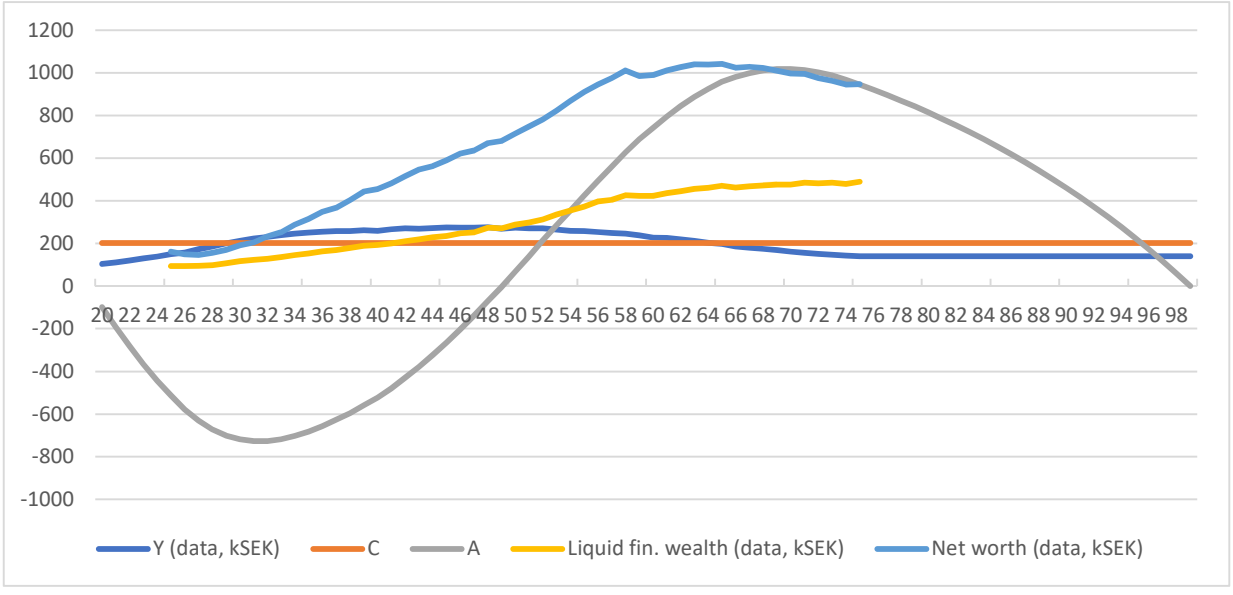
80% replacement rate in retirement – cuts retirement savings by half



Data on household disposable income in 2000, 25–75 years



Data on liquid financial wealth and net worth relative to model's A



Outline of day 1

- Solving the sequential formulation of a simple consumption-savings problem
- Dynamic programming formulation of a "realistic" consumer's problem (Carroll, QJE 1997)
  - No-borrowing constraint / tight borrowing constraint
  - Uninsurable income risk (Carroll and Samwick, JME 1997)
- General tips and advice for programming:
  - Pros and cons on programming languages
  - The coding process
  - Books, resources, etc
- A quasi-algorithm for solving and simulating the consumer's problem using the endogenous grid point method (Carroll, EL 2006)
- Discussion of the problem set

Introducing a borrowing constraint

$$U = \max_{\{C_t, A_t\}_{t=1}^{T}} \sum_{t=1}^{T} \beta^{t-1} u(C_t) \tag{15}$$

subject to

$$C_1 + A_1 = Y_1 + \hat{A} \tag{16}$$

$$C_t + A_t = Y_t + A_{t-1} R \qquad t = 2, 3, ..., T \tag{17}$$

$$A_t \geq \phi \qquad t = 1, 2, 3, ..., T \tag{18}$$

$$A_T = 0 \tag{19}$$

$\phi = 0$: no borrowing constraint

Introducing stochastic labor income

$$U = \max_{\{C_t, A_t\}_{t=1}^{T}} \mathbb{E}_0 \sum_{t=1}^{T} \beta^{t-1} u(C_t) \tag{20}$$

subject to

$$C_1 + A_1 = \underbrace{Y_1 + \hat{A}}_{=X_1} \tag{21}$$

$$C_t + A_t = \underbrace{Y_t + A_{t-1}R}_{=X_t} \qquad t = 2, 3, ..., T \tag{22}$$

$$A_t \geq 0 \qquad t = 1, 2, 3, ..., T \tag{23}$$

$$A_T = 0 \tag{24}$$

where Y_t follows a stochastic process.

The income process of Carroll and Samwick (1997)

Let $y_{it} = \ln(Y_{it})$. Then for $t \leq 45$:

$$\begin{aligned} y_{it} &= g_t + z_{it} + \omega_{it}, && (25) \\ z_{it} &= \rho z_{it-1} + \eta_{it}, && (26) \\ y_{it} &\geq \ln(\underline{Y}). && (27) \end{aligned}$$

where g_t is a deterministic life-cycle trend, $\rho \in (0, 1]$, $\underline{Y} \geq 0$. Notation: $Y_{it} = Y_{it}^p \exp(\omega_{it})$. Retirement:

$$\begin{aligned} Y_{i,t} &= \lambda Y_{i,45}^p \\ &= \lambda \exp(g_{45} + z_{i45}), \quad t > 45. && (28) \end{aligned}$$

Stochastics:

$$z_{i1} \sim N\left(-\sigma_z^2/2, \sigma_z^2\right). \quad (29)$$

$$\eta_{it} \sim N\left(-\sigma_\eta^2/2, \sigma_\eta^2\right). \quad (30)$$

$$\omega_{it} \sim N\left(-\sigma_\omega^2/2, \sigma_\omega^2\right). \quad (31)$$

In the last 10-15 years, people have documented that log-normal income risk does not fit higher-order moments of the data – big literature.

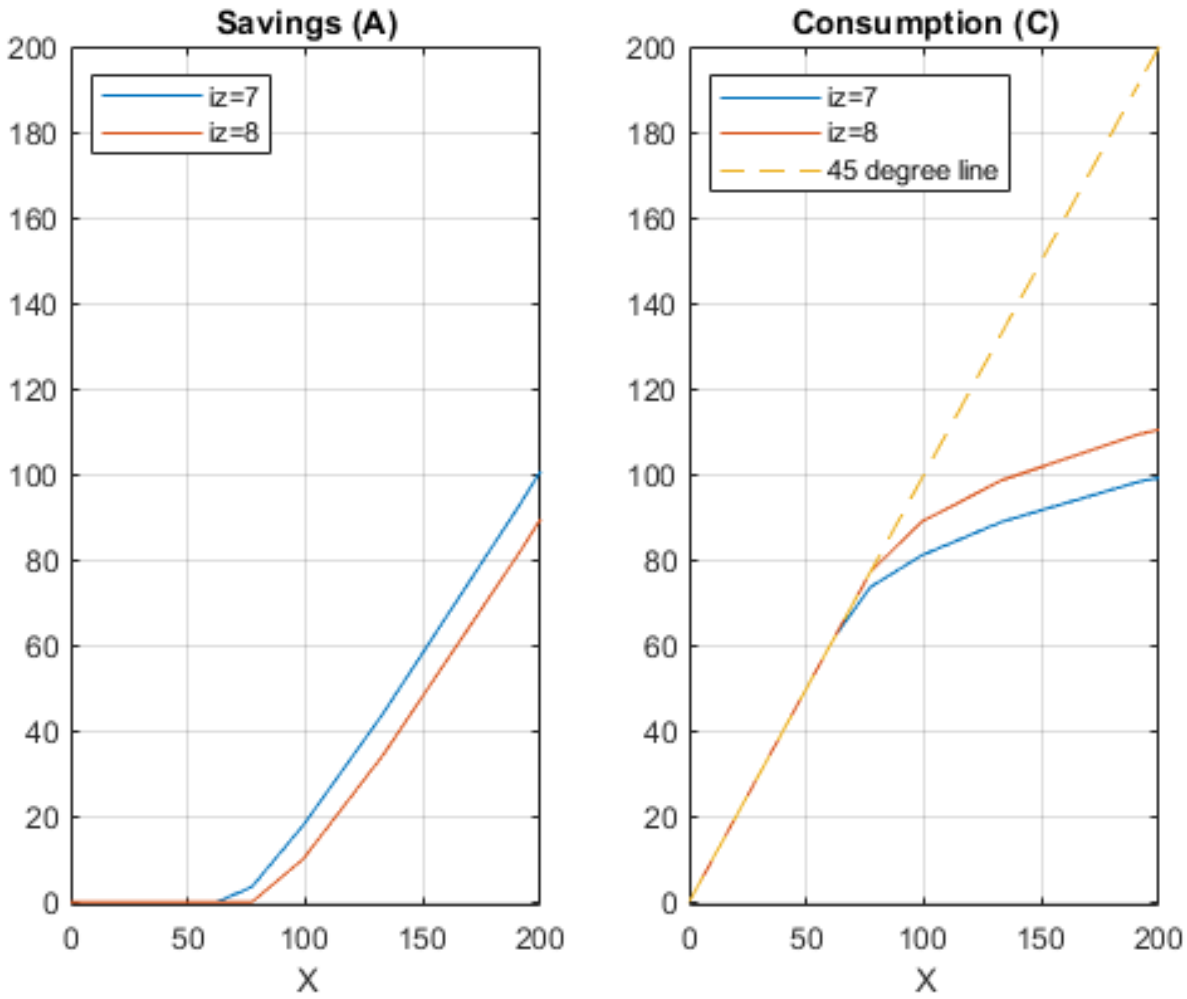
Market incompleteness introduces numerical challenges

- Uninsurable income risk (ω_{it}, z_{it}) implies a precautionary savings motive, in addition to life-cycle motive. Omit subscript i for notational convenience.
- $A_t \geq 0$ implies $C_t \approx Y_t$ early in life, in particular if $X_t \leq Y_t^p$ (PIH logic).

$$u'(C_t) \geq \beta R \, \mathbb{E} \, u'(C_{t+1}) \quad t = 1, 2, \ldots T-1 \tag{32}$$

- Kink in policy function for optimal savings, $A_t(z_t, X_t)$, and consumption $C_t(z_t, X_t)$.

Policy functions in Kilström, Sigurdsson, and Vestman (2022)



Casting the consumer's problem as a dynamic programming problem

Let $\mathcal{S}_t = \{z_t, X_t\}$ denote the state variables for $t = 1, ..., T$. Budget constraints and laws of motions for state variables:

$$C_t + A_t = X_t, \tag{33}$$

$$A_t \geq 0, \tag{34}$$

where X_t denotes cash on hand. Its law of motion is:

$$X_{25} = \hat{A}_{25} + Y_{25}, \tag{35}$$

where $\hat{A}_{25}$ is initial financial wealth.

$$X_{t+1} = A_t R + Y_{t+1} \tag{36}$$

$$= A_t R + e^{g_{t+1} + \rho z_t + \eta_{t+1} + \omega_{t+1}} \quad t = 25, ..., 63 \tag{37}$$

$$X_{t+1} = A_t R + Y_{t+1}$$

$$= A_t R + \lambda e^{g_{64} + z_{t+1}} = A_t R + \lambda e^{g_{64} + z_{64}} \quad t = 64, ..., 98 \tag{38}$$

For $t < 64$, the law of motion for z_t is given by (26). For $t \geq 64$, $z_{t+1} = z_t$.

Dynamic programming problem

The optimization problem is given by:

$$V_t(\mathcal{S}_t) \quad = \quad \max_{C_t, A_t} \quad u\left(C_t\right) + \beta\, \mathbb{E}_t\left[V_{t+1}(\mathcal{S}_{t+1})\right], \tag{39}$$

subject to (25)-(31) and (33)-(36), and where $V_{T+1} = 0$. Notice that $\mathbb{E}_t[.]$ is short-hand for a conditional expectations operator: $\mathbb{E}[.|\mathcal{S}_t]$.

3 methodological challenges

Need to:

1. numerically approximate the functions $\{V_t(\mathcal{S}_t), A_t(\mathcal{S}_t), C_t(\mathcal{S}_t)\}_{t=1}^{T}$.
2. maximize $V_t(z_t, X_t)$ in the admissable range of values for $C_t(.)$ and $A_t(.)$ using an optimizer.
3. perform integration to obtain $\mathbb{E}\left[V_{t+1}(\mathcal{S}_{t+1})|\mathcal{S}_t\right]$, given laws of motion for the state variables and given the control variable $A_t(.)$.

Outline of day 1

- Solving the sequential formulation of a simple consumption-savings problem
- Dynamic programming formulation of a "realistic" consumer's problem (Carroll, QJE 1997)
  - No-borrowing constraint / tight borrowing constraint
  - Uninsurable income risk (Carroll and Samwick, JME 1997)
- General tips and advice for programming:
  - Pros and cons on programming languages
  - The coding process
  - Books, resources, etc
- A quasi-algorithm for solving and simulating the consumer's problem using the endogenous grid point method (Carroll, EL 2006)
- Discussion of the problem set

General tips – choice of programming language

- Matlab: easy to use and debug, low threshold. Not the fastest language but perhaps the right language to start with.
- Julia: the future of scientific computing? open source. Still immature. Your current code will not run in 5–10 years?
- Python: open source, lots of existing code. Not specialized towards high performance computing.
- Fortran: still the fastest but old and cumbersome coding and debugging process. Choice of compiler matters. Intel Fortran expensive. Not the right choice anymore.

See e.g.
`https://www.sas.upenn.edu/~jesusfv/Update_March_23_2018.pdf`
`https://cepr.org/voxeu/columns/`
`which-programming-language-best-economic-research-julia-matlab-python-`

General tips – how to make coding sessions efficient

- Use Github or equivalent tool for version control.
- Make it a habit to commit changes at the end of every coding session.
- Keep a simple Word file where you at the end of every session write down what the next steps are.
- The quality / time trade-off: it is useful to sit two together when coding. Limits round-abouts, reduces bugs, and makes code more transparent.
- Do not invent the wheel. Depending on your choice of language, there will be different kinds of in-built functions for integration, interpolation, optimization, etc. Do not code your own functions unless they do not exist. Clearly beneficial to look around for existing code published by people you trust.
- Replication packages at top journal is a fairly new resource.
- Comment your code extensively and your future self will thank you.

General tips – how to make the code efficient

- Use a profiler
- Learn how to parallelize your code to utilize all cores and clusters. E.g., OpenMP for Fortran and C++.
- Each language requires some tricks: vectorization, whether matrices are stored column-wise or row-wise in the cache memory (read up on so called "cache misses").
- Make sure your compiler optimizes your code.

Choice of hardware

- For developing code it is most efficient to use your laptop or desktop
- But figure out how to scale your hardware resources
- Most universities have high performance computing centers (clusters) – and they love to have users from the social sciences
- I have started to use Uppmax: `https://www.uppmax.uu.se/`
- Uppmax staff offer great support and courses
- PDC at Royal Institute of Technology is another option: `https://www.pdc.kth.se/`
- User accounts at these national resources are administered here where you apply: `https://supr.naiss.se/`
- Requires a little bit of investment in Linux and the scheduler (e.g. SLURM)

Outline of day 1

- Solving the sequential formulation of a simple consumption-savings problem
- Dynamic programming formulation of a "realistic" consumer's problem (Carroll, QJE 1997)
  - No-borrowing constraint / tight borrowing constraint
  - Uninsurable income risk (Carroll and Samwick, JME 1997)
- General tips and advice for programming:
  - Pros and cons on programming languages
  - The coding process
  - Books, resources, etc
- A quasi-algorithm for solving and simulating the consumer's problem using the endogenous grid point method (Carroll, EL 2006)
- Discussion of the problem set

Applying the endogenous grid point method to the problem

The optimization problem is given by:

$$V_t(\mathcal{S}_t) \quad = \quad \max_{C_t, A_t} \quad u\left(C_t\right) + \beta\, \mathbb{E}_t\left[V_{t+1}(\mathcal{S}_{t+1})\right], \tag{40}$$

subject to (25)-(31) and (33)-(36), and $V_{T+1} = 0$.
To Do:

- Derive the Euler equation
- Choose integration method for the stochastic income during working life
- Choose approximation method for $A_t(\mathcal{S}_t)$ and $C_t(\mathcal{S}_t) = X_t(\mathcal{S}_t) - A_t(\mathcal{S}_t)$.

Euler equation

The optimization problem is given by:

$$V_t(\mathcal{S}_t) \quad = \quad \max_{A_t} \quad u\left(X_t - A_t\right) + \beta\, \mathbb{E}_t\left[V_{t+1}(\mathcal{S}_{t+1})\right], \tag{41}$$

subject to (25)-(31) and (34)-(36), and $V_{T+1} = 0$.
FOC wrt A_t:

$$u'(X_t - A_t) \geq \beta\, \mathbb{E}_t\left[\frac{dV_{t+1}(\mathcal{S}_{t+1})}{dX_{t+1}} \cdot \frac{\partial X_{t+1}}{\partial A_t}\right] \tag{42}$$

$$= \beta\, \mathbb{E}_t\left[\frac{dV_{t+1}(\mathcal{S}_{t+1})}{dX_{t+1}} \cdot R\right] \tag{43}$$

$$\tag{44}$$

Envelope condition:

$$\frac{dV_t(\mathcal{S}_t)}{dX_t} = \frac{\partial V_t(\mathcal{S}_t)}{\partial X_t} = u'(X_t - A_t) \tag{45}$$

i.e., $\frac{\partial V_t(\mathcal{S}_t)}{\partial A_t} = 0$

Euler equation

Lead forward and substitute:

$$u'(X_t - A_t) \geq \beta R \, \mathbb{E}_t \left[u'(X_{t+1} - A_{t+1}) \right] \tag{46}$$

For working life, i.e. t=25,...,63:

$$u'(X_t - A_t) \geq \beta R \, \mathbb{E}_t \left[u'(A_t R + e^{g_{t+1} + \rho z_t + \eta_{t+1} + \omega_{t+1}} - A_{t+1}(\mathcal{S}_{t+1})) \right] \tag{47}$$

For retirement, i.e. t=64,...,98:

$$u'(X_t - A_t) \geq \beta R u'(A_t R + \lambda e^{g_{64} + z_{64}} - A_{t+1}(\mathcal{S}_{t+1})) \tag{48}$$

Notice that if $A_t > 0$ ($A_t > \phi$) then the Euler equations holds with equality.

Computing expectations – two main options

For ω_{t+1} Guassian quadrature is the best. For η_{t+1} there are two options.

1. Quadrature for η_{t+1}
2. Discretize $z_{t+1}|z_t$ to a Markov process as in e.g. Tauchen (1986), or Rouwenherst (1995), or Flodén (2008).
   - `https://martinfloden.net/files/ar1_processes_matlab_code.zip`
   - `https://giuliofella.net/research/code/code.html`

Quadrature

E.g. Judd 1998, p. 262:

$$\mathbb{E}[\eta_{t+1}] \approx \sum_{i=1}^{n} \left(\mu_\eta + w_i \cdot \sigma_\eta \cdot q_i\right) = \sum_{i=1}^{n} \left(-\sigma_\eta^2/2 + w_i \cdot \sigma_\eta \cdot q_i\right) \tag{49}$$

$$\begin{aligned}
&\mathbb{E}_t\left[u'(A_t R + e^{g_{t+1}+\rho z_t+\eta_{t+1}} - A_{t+1}(A_t R + e^{g_{t+1}+\rho z_t+\eta_{t+1}}, z_{t+1}))\right] \\
\approx &\pi^{-1/2} \sum_{i=1}^{n} \left(w_i \cdot u'(A_t R + e^{g_{t+1}+\rho z_t-\sigma_\eta^2/2+\sqrt{2}\sigma_\eta \cdot q_i} - A_{t+1}(A_t R + e^{g_{t+1}+\rho z_t-\sigma_\eta^2/2+\sqrt{2}\sigma_\eta \cdot q_i}, \rho z_t - \sigma_\eta^2/2 + \sqrt{2}\sigma_\eta \cdot q_i))\right)
\end{aligned} \tag{50}$$

See also: `https://en.wikipedia.org/wiki/Gauss-Hermite_quadrature`

A Markov chain for z_t using the Tauchen or Roewenherst method

- $z_t = \rho z_{t-1} + \eta_t = \rho^2 z_{t-2} + \rho\eta_{t-1} + \eta_t = ...$
- At each t, compute the unconditional distribution $var(z_t|z_0) = \sigma_\eta^2 + \rho^2\sigma_\eta^2 + ... = \sum_{j=0}^{t-1} \rho^{2\cdot j}\sigma_\eta^2$
- Choose number of discrete states for z_t, N_z
- Choose grid points (values) for z_0
- Choose grid points (values) for z_t
- Compute transition propabilities $\pi(z_{k,t+1}|z_{j,t})$. Notice that $\sum_{k=1}^{N_z} \pi(z_{k,t+1}|z_{j,t}) = 1$ for all j, all t.
- For instance Guillo Fella has Matlab code for this: `https://giuliofella.net/research/code/code.html`

Integrating over both shocks

If z_t discretized then:

- $\mathbb{E}[u'(C_{t+1})|(X_t, z_{j,t})] = \sum_{k=1}^{N_z} \sum_{i}^{N} w_i \cdot \pi(z_{k,t+1}|z_{j,t}) \cdot u'(C_{t+1}(X_{t+1}, z_{t+1}))$
- z_t always on grid defined by previously mentioned methods.
- In simulation, draw uniform random variable to determine transition of $z_{t+1}|z_t$

If z_t continuous and integration over η then:

- $\mathbb{E}[u'(C_{t+1})|(X_t, z_{j,t})] = \sum_{k=1}^{N} \sum_{i}^{N} w_i \cdot w_k \cdot u'(C_{t+1}(X_{t+1}, z_{t+1}))$.
- Need grid for z_t, $\{z_{1,t}, z_{2,t}, ..., z_{N_z,t}\}$ (varying with age).
- In simulation, draw normal random variable for η_t to determine $z_{t+1}|z_t$.
- In both solution and simulation, need to interpolate/extrapolate between the grid points for z_t.

Notice: this only applies to the working life phase $t = 25, 26, .., 63$.

Approximating functions

- The most robust and simple method is piece-wise linear interpolation/extrapolation of $A_t(X_t, z_t)$, and $V_t(X_t, z_t)$ if necessary to have it. Matlab function: interp1, interp2, interp3.
- The univariate case: $f(x) \approx f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - xi} \times (x - x_i)$, where $x \in [x_i, x_{i+1}]$.
- The function will be continuous and monotone but not differentiable everywhere. Not necessarily a problem.
- How to handle extrapolation might require some thought.
- Approximating functions using polynomials or other kinds of splines is big area in itself.

The endogenous grid point method of Carroll (2006)

- Has become the standard method.
- We do not maximize V_t subject to constraints using some kind of optimization algorithm.
- Instead, we rely on the Euler equation (FOCN). In particular, we construct a grid for A_t and back out the corresponding value for cash-in-hand, $X_t(A_t, z_t)$.
- If $T = \infty$ then optimization on V and iterating recursively (replacing continuation value with latest approximation) is a contraction mapping, but iterating on the Euler equation is not.
- Going forward, assume $u(c) = \frac{c^{1-\gamma}}{1-\gamma}$ for simplicity, so that $u'(c) = c^{-\gamma}$.

Step 1 – declare objects

Define the following objects:

- Declare a grid for savings $\mathcal{A} = \{a^1, a^2, ..., a^{N_a}\}$ where $a^1 = 0$, or $a^1 = \phi$.
- Set up an integration procedure for ω, (w_i, q_i), $i = 1, ..., N_q$ (quadrature)
- Set up an integration procedure for $z_{t+1}|z_t$ and so that you have a grid $\mathcal{Z}_t = \{z_t^1, z_t^2, ...z_t^{N_z}\}$ and transition probabilities π_t for $t = 25, 26, ..., T-1$. Notice that the grid is identical for $t = 64, 65, ..., T-1$ and the transition propabilities are trivial/unnecessary. (Rouwenherst or similar)
- Declare an exogenous grid for cash-on-hand at $t = T$: $\mathcal{X} = \{x^1, ..., x^{N_x}\}$.
- Define $\mathcal{A} \times \mathcal{Z}_t = \hat{\mathcal{S}}_t$ for $t = 1, ..., T$.
- Define how many grid points for cash-on-hand that should cover the range $[(\underline{Y}), X(a^1))$: $Nx_{constrained}$. Notice that unless you need the value function in your analysis, $Nx_{constrained} = 1$ is sufficient. The reason is that that policy functions are linear in this range whereas the value function has curvature.
- Notice that there are many numerical parameters, in addition to the model's parameters: $N_a, N_z, N_q, ..$

Step 2 – solve by backward induction in the retirement phase

- Set $t = T$.
  - Let $A_T(x^i, z^{T,k}) = 0$ for all i and k. $C_T = x^i$
  - Compute the value function if it is necessary for your analysis: $V_T(x^i, z^{T,k}) = (x^i)^{1-\gamma}/(1-\gamma)$.
- Set $t = T - 1$.
  - For each grid point $(a, z_t) \in \hat{\mathcal{S}}_t$ solve for $X_t(z_t, a)$ using equation (48):

$$X_t(a, z_t) = a + (\beta R)^{-1/\gamma} \cdot (aR + \lambda e^{g_{64}+z_t} - A_{t+1}(aR + \lambda e^{g_{64}+z_t}, z_t)) \quad (51)$$

  - We have the policy function $A_t(\mathcal{S}_t)$ for the unconstrained range of X: $[X_t(a^1, z_t), X_t(a^{N_a}, z_t)]$. Augment this function with constrained region $A_t(., z_t) = 0$. So the total dimension of the policy function is $(N_a + Nx_{constrained}) \times N_z$. If the value function is not necessary then: $(N_a + 1) \times N_z$ where $A_t(\underline{Y}, z_t) = 0$ is the added element. Set $C_t(x^i, z_t)) = x^i - A_t(x^i, z_t)$.
  - Repeat for $t = T - 2, ..., 64$.

Step 3 – solve by backward induction in the working life phase

- Set $t = 63$.
  - For each grid point $(a, z_t) \in \hat{\mathcal{S}}_t$ solve for $X_t(z_t, a)$ using equation (47):

$$X_t(a, z_{j,t}) = a + (\beta R)^{-1/\gamma} \cdot \left(\sum_{k=1}^{N_z} \sum_{i}^{N_q} w_i \cdot \pi(z_{k,t+1}|z_{j,t}) \cdot (C_{t+1}(X_{t+1}, z_{t+1}))^{-\gamma} \right)^{-1/\gamma} \tag{52}$$

  - We have the policy function $A_t(\mathcal{S}_t)$ for the unconstrained range of X: $[X_t(a^1, z_t), X_t(a^{N_a}, z_t)]$. Augment this function with $A_t(., z_t) = 0$ as in the retirement phase.
  - Repeat for $t = 62, 61, ..., 25$.

Concrete programming tips

- Separate model parameters from technical parameters – you want flexilibity also on the technical ones
- Construct the objects necessary for integration just once, before the solution algorithm starts
- Define subroutines that represent the laws of motions for X and z: $X_{t+1}(a, z_{j,t})$ and $z_{t+1}(z_{j,t})$. Notice that they return vectors $(N_z \cdot N_q) \times 1$ and each outcome is associated with probability $w_i \cdot \pi(z_{k,t+1}|z_{j,t})$.
- The simulation:
  - Use the same set of shocks every time to be able to debug efficiently. Draw them once, then save them and then load them every time.
  - Alternatively, set a seed in the random number generator but make sure it is set before every run of your code.
  - Re-use as much of the code from the solution as possible, for instance the law of motions for the state variables.
  - Make the problem minimalistic first: no need to start with large T, or with dense grids, or with income uncertainty.
  - Play around with model parameters for which you have a strong intuition, e.g. $\beta R = 1$, $\phi << 0$, and $\sigma_\eta = \sigma_\omega = 0$

Outline of day 1

- Solving the sequential formulation of a simple consumption-savings problem
- Dynamic programming formulation of a "realistic" consumer's problem (Carroll, QJE 1997)
  - No-borrowing constraint / tight borrowing constraint
  - Uninsurable income risk (Carroll and Samwick, JME 1997)
- General tips and advice for programming
  - Pros and cons on programming languages
  - The coding process
  - Books, resources, etc
- A quasi-algorithm for solving and simulating the consumer's problem using the endogenous grid point method (Carroll, EL 2006)
- Discussion of the problem set

The problem set – useful resources

- Giulio Fella: `https://giuliofella.net/`
- Jesus Fernandez-Villaverde:
  `https://www.sas.upenn.edu/~jesusfv/research.html`
- Lecture 2 of Ben Moll's first-year course: `https://benjaminmoll.com/wp-content/uploads/2021/04/Lecture2_EC442_Moll.pdf`
- Ben Moll and Greg Kaplan have a lot of code:
  `http://benjaminmoll.com/ha_codes/`
  - Have a look at the readme.txt file and egp_IID_lifecycle.m if you want
- Lot's of other code, e.g. Chris Caroll:
  `http://econ.jhu.edu/People/CCarroll/EndogenousArchive.zip`
- Quantitative Methods in Macroeconomics at Stockholm University