Financial Theory IV: Solving Structural Models

Dan Greenwald

Spring 2024

Dynamic Programming

- Let x_t be endogenous states, z_t be exogenous states, and y_t be controls.
- Basic problem:

$$\begin{aligned} V(x_t, z_t) &= \max_{y_t \in \Gamma(x_t, z_t)} F(x_t, y_t, z_t) + \beta E_t \left[V(x_{t+1}, z_{t+1}) \right] \\ x_{t+1} &= g(x_t, y_t, z_t) \\ z_{t+1} &= h(z_t, \varepsilon_{t+1}) \end{aligned}$$

- Example: consumption-savings problem.

$$\begin{aligned} V(a_t, w_t) &= \max_{x_t \geq 0} u\left(a_t + w_t \bar{L} - s_t\right) + \beta E_t \left[V(a_{t+1}, w_{t+1}) \right] \\ a_{t+1} &= R s_t \\ \log w_{t+1} &= (1 - \rho) \log \bar{w} + \rho \log w_t + \varepsilon_{t+1}, \qquad \varepsilon_{t+1} \sim N(0, \sigma^2) \end{aligned}$$

Dynamic Programming: Discrete Models

Discrete Dynamic Programming

- Very simple and robust approach: assume $x_t \in \mathcal{X} = (\bar{x}_1, \ldots, \bar{x}_N)$, $z_t \in \mathcal{Z} = (\bar{z}_1, \ldots, \bar{z}_K)$.
  - Easy to estimate time series using Hamilton filter (see Farmer, 2017).

- Basic problem reframed:

$$V(x_t, z_t) = \max_{x_{t+1} \in \Gamma(x_t, z_t)} F(x_t, z_t, x_{t+1}) + \beta \sum_{z_{t+1}} P(z_{t+1} | z_t) V(x_{t+1}, z_{t+1})$$

- Effects of discretization:
  - Choose x_{t+1} directly instead of y_t (can't leave grid).
  - Expectation is matrix multiplication.

- Notation: $\mathbf{X}(a, b; c)$ is a matrix where the columns stack over a and b (i.e., $(a_1, b_1), (a_1, b_2), \ldots, (a_2, b_1), \ldots)$ and the rows stack over c.

Discrete Dynamic Programming

- **Step 1:** given iteration k guess $\mathbf{V}_k$, optimize decision.
- Define

$$\mathbf{Q}(x_t, z_t; x_{t+1}) = \underbrace{\mathbf{F}(x_t, z_t; x_{t+1})}_{NK \times N} + \beta \Big(\underbrace{\mathbf{P}(z_t; z_{t+1})}_{K \times K} \otimes \underbrace{\mathbf{1}_N}_{N \times 1} \Big) \underbrace{\mathbf{V}(x_{t+1}; z_{t+1})'}_{K \times N}$$

  if $x_{t+1} \in \Gamma(x_t, z_t)$, and $-\infty$ otherwise.
- Reminder: $\otimes$ is the Kronecker product, so that

$$A \otimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{n1}B & \cdots & A_{nn}B \end{bmatrix}$$

- Define $x^*_{t+1}(x_t, z_t) = \arg\max_{x_{t+1}} \mathbf{Q}(x_t, z_t; x_{t+1})$. This is the column-wise max.

Discrete Dynamic Programming

- **Step 2:** given decisions, update $\mathbf{V}$ ("Howard Improvement").
  - Can update $\mathbf{V}_{k+1}$ by plugging in x^*_{t+1} and $\mathbf{V}_k$ on the RHS, then iterate, but this is slow.
  - Better approach: solve for **exact** value function under policy x^*_{t+1}.

- Define:

$$A(x_t, z_t, x_{t+1}, z_{t+1}) = P(z_{t+1}|z_t) \cdot \mathbf{1}\left\{x_{t+1} = x^*_{t+1}(x_t, z_t)\right\}$$
$$F^*(x_t, z_t) = F(x_t, z_t, x^*_{t+1})$$

- Then we have:

$$\underbrace{\mathbf{V}(x_t, z_t)}_{NK \times 1} = \underbrace{\mathbf{F}^*(x_t, z_t)}_{NK \times 1} + \beta \underbrace{\mathbf{A}(x_t, z_t; x_{t+1}, z_{t+1})}_{NK \times NK} \underbrace{\mathbf{V}(x_{t+1}, z_{t+1})}_{NK \times 1}$$

which implies the exact solution $\mathbf{V}_{k+1} = (I - \beta \mathbf{A})^{-1} \mathbf{F}^*$.

Discrete Dynamic Programming

- Iterate on Steps 1 and 2 until x^*_{t+1} stops changing. Then you are done!
- Stationary distribution: eigenvector of $\mathbf{A}'$ associated with unit eigenvalue.
  - Similarly, stationary dist. of exogenous states is eigenvector of P' with unit eigenvalue.
- Note: **A** will contain many zeros, often better to use sparse matrices.
- For P, use Rouwenhorst (1995) method to approximate Gaussian AR(1) processes.
  - Other approximations struggle as $\rho \to 1$.
  - Better to read treatment in Kopecky and Suen, RED 2010.
- Suffers from curse of dimensionality, but GPUs can provide huge speedup!

Special Case: Exogenous Asset Pricing

- Assume that all states are exogenous.
- Combine $E_t[M_{t+1}R_{t+1}] = 1$ and definition $R_{t+1} = (P_{t+1} + D_{t+1})/P_t$ to obtain

$$PD(z_t) = E_t\left\{M(z_t, z_{t+1})\left(PD(z_{t+1}) + 1\right)\frac{D(z_{t+1})}{D(z_t)}\right\}.$$

- Then we can solve for PD exactly with a **single linear inversion**:

$$A(z_t, z_{t+1}) \equiv P(z_t, z_{t+1})M(z_t, z_{t+1})\frac{D(z_{t+1})}{D(z_t)}$$

$$\mathbf{PD}(z_t) = \mathbf{A}(z_t; z_{t+1})\left(\mathbf{PD}(z_{t+1}) + \mathbf{1}_K\right)$$

$$\mathbf{PD} = (I - \mathbf{A})^{-1}\mathbf{1}_K$$

Dynamic Programming: Continuous Models

Generic Optimality Conditions

- As long as the problem is well-behaved (uniquely determined by FOCs), it is usually better to solve the FOCs than to directly use the value function.
- Typical approach is to just start taking derivatives, but can actually be more systematic.
- Let's add some additional structure (slight change of notation):
  - Let c_t be consumption, and y_t be all **other** controls.
  - Let $\Psi(x_t, c_t, y_t, z_t) \geq 0$ be the budget constraint, and $\Gamma(x_t, c_t, y_t, z_t) \geq 0$ be all **other** constraints.
  - Assume the budget constraint is written $c_t \leq \cdots$ so that $\partial \Psi_t / \partial c_t = -1$.
  - Let $F(x_t, y_t, z_t) = u(x_t, c_t, y_t, z_t)$.

Generic Optimality Conditions

- Generic optimality condition for y_t:

$$0 = \underbrace{\left(\frac{\partial u_t}{\partial c_t}\right)^{-1} \frac{\partial u_t}{\partial y_t}}_{\text{utility}} + \underbrace{\frac{\partial \Psi_t}{\partial y_t}}_{\text{resources}} + \underbrace{\mu_t \frac{\partial \Gamma_t}{\partial y_t}}_{\text{constraints}} + \underbrace{\Omega_t \frac{\partial x_{t+1}}{\partial y_t}}_{\text{continuation}}$$

- All quantities expressed in units of consumption.
- Marginal continuation values Ω_t defined by fixed point

$$\Omega_t = E_t \left\{ M_{t+1} \left[\left(\frac{\partial u_{t+1}}{\partial c_{t+1}}\right)^{-1} \frac{\partial u_{t+1}}{\partial x_{t+1}} + \frac{\partial \Psi_{t+1}}{\partial x_{t+1}} + \mu_{t+1} \frac{\partial \Gamma_{t+1}}{\partial y_{t+1}} + \Omega_{t+1} \frac{\partial x_{t+2}}{\partial x_{t+1}} \right] \right\}$$

where M_{t+1} is the SDF. **Note: works for EZW preferences.**

Example: Kaltenbrunner and Lochstoer (2010)

- Preferences: $U_t = \left((1-\beta) C_t^{1-\rho} + \beta E_t \left[U_{t+1}^{1-\gamma} \right]^{\frac{1-\rho}{1-\gamma}} \right)^{\frac{1}{1-\rho}}$
- Budget constraint: $C_t \leq Z_t^{1-\alpha} K_t^{\alpha} - i_t K_t$.
- Endogenous state LOM: $K_{t+1} = (1-\delta) K_t + \phi(i_t) K_t$.
- Exogenous state LOM: $\log Z_{t+1} = \phi \log Z_t + \varepsilon_{t+1}$.
- Optimality conditions:

$$0 = -1 + \phi'(i_t) \Omega_{K,t}$$

$$\Omega_{K,t} = E_t \left\{ M_{t+1} \left[\alpha \left(\frac{Z_{t+1}}{K_{t+1}} \right)^{1-\alpha} - i_{t+1} + \Big((1-\delta) + \phi(i_{t+1}) \Big) \Omega_{K,t+1} \right] \right\}$$

Example: Kaltenbrunner and Lochstoer (2010)

- Preferences: $U_t = \left((1-\beta) C_t^{1-\rho} + \beta E_t \left[U_{t+1}^{1-\gamma} \right]^{\frac{1-\rho}{1-\gamma}} \right)^{\frac{1}{1-\rho}}$
- Budget constraint: $C_t \leq Z_t^{1-\alpha} K_t^{\alpha} - i_t K_t.$
- Endogenous state LOM: $K_{t+1} = (1-\delta) K_t + \phi(i_t) K_t.$
- Exogenous state LOM: $\log Z_{t+1} = \phi \log Z_t + \varepsilon_{t+1}.$
- Optimality conditions:

$$1 = E_t \left[M_{t+1} R_{t+1} \right]$$

$$R_{t+1} \equiv \frac{\alpha (Z_{t+1}/K_{t+1})^{1-\alpha} - i_{t+1} + \Big((1-\delta) + \phi(i_{t+1}) \Big) q_{t+1}}{q_t}$$

$$q_t \equiv \phi'(i_t)^{-1}$$

Complementary Slackness

- Complementary slackness: given constraint Γ_t and multiplier μ_t:

$$\mu_t \Gamma_t = 0, \qquad \mu_t \geq 0, \qquad \Gamma_t \geq 0.$$

- Example: lower bound $y_t \geq 0$.
  - Challenge: kinked, nondifferentiable policy function.
- **Auxiliary variable** (Garcia and Zangwill) approach:
  - Define policy function as auxiliary variable α_t.
  - Define $y_t = \max(\alpha_t, 0)^k$ for $k > 1$.
  - Define $\mu_t = \max(-\alpha_t, 0)^k$ for $k > 1$.
- Delivers continuously differentiable policy function.

Time Iteration

- Assume equilibrium conditions follow $f\left(x, y, z, \mathcal{E}(x, y, z)\right) = 0$, where

$$\mathcal{E}_t = E_t\left[q(x_{t+1}, y_{t+1}, z_{t+1})\right]$$

- Choose grid $\{\bar{x}_i, \bar{z}_i\}$ and basis functions $\psi(s, z)$.
- Let b_k be the coefficients from the previous (kth) iteration.
- Key idea: use previous guess b_k to form expectations $\mathcal{E}$:

$$\begin{aligned} y_{t+1} &= b_k' \psi(x_{t+1}, z_{t+1}) \\ \mathcal{E}_t &= \sum_j \omega_j q(x_{t+1}, y_{t+1}, z_{t+1}) \end{aligned}$$

  where ω_j are quadrature weights, then solve for y_t given $\mathcal{E}_t$.

Time Iteration on Controls

- **Time iteration on controls**: for each $(\bar{x}_i, \bar{z}_i)$, choose y_i so $f\left(\bar{x}_i, y_i, \bar{z}_i, \mathcal{E}\left(\bar{x}_i, \bar{z}_i; b_k\right)\right) = 0$, with

$$\mathcal{E}_t(\bar{x}_i, \bar{z}_i; b_k) = \sum_j \omega_j q\Big(x_{t+1}, b_k' \psi(x_{t+1}, z_{t+1}), z_{t+1}\Big)$$

- Recipe: given y_i, quadrature node $\bar{\varepsilon}_j$, compute

$$\begin{aligned} x_{t+1} &= g(\bar{x}_i, y_i, \bar{z}_i) \\ z_{t+1} &= h(\bar{z}_i, \bar{\varepsilon}_j). \end{aligned}$$

- Use nonlinear equation solver. Gradient:

$$\frac{df}{dy_t} = \frac{\partial f}{\partial y_t} + \frac{\partial f}{\partial \mathcal{E}_t} E_t \left[\frac{\partial x_{t+1}}{\partial y_t} \left(\frac{\partial q}{\partial x_{t+1}} + \frac{\partial q}{\partial y_{t+1}} \frac{\partial y_{t+1}}{\partial x_{t+1}} \right) \right]$$

  where $\partial y_{t+1} / \partial x_{t+1} = b_k' (\partial \psi(x_{t+1}, z_{t+1}) / \partial x_{t+1})$.

- Once solutions $\{y_i^*\}$ have been found, choose b_{k+1} so that $y_i^* = b_{k+1}' \psi(\bar{x}_i, \bar{z}_i)\ \forall i$, repeat.

Time Iteration

- Iterating on controls is straightforward but not very efficient.
  - Additional challenge: y_{t+1} may not be non-smooth in states.

- Alternative: **time iteration on coefficients**. Solve directly for $\hat{b}$ that satisfies

$$f\left(\bar{x}_i, \hat{b}'\psi(\bar{x}_i, \bar{z}_i), \bar{z}_i, \mathcal{E}\right) = 0.$$

  i.e., compute $y_i = \hat{b}'\psi(\bar{x}_i, \bar{z}_i)$ and proceed as before.

- Use nonlinear equation solver with gradient

$$\frac{\partial f}{\partial \hat{b}} = \left\{ \frac{\partial f}{\partial y_t} + \frac{\partial f}{\partial \mathcal{E}_t} E_t \left[\frac{\partial x_{t+1}}{\partial y_t} \left(\frac{\partial q}{\partial x_{t+1}} + \frac{\partial q}{\partial y_{t+1}} \frac{\partial y_{t+1}}{\partial x_{t+1}} \right) \right] \right\} \frac{\partial y_t}{\partial \hat{b}}$$

  where $\partial y_t / \partial \hat{b} = \psi(x_t, z_t)'$.

- Update $b_{k+1} = \hat{b}$ and repeat until $||b_{k+1} - b_k||$ is smaller than some threshold.

Direct Solution

- Most efficient (but least robust): solve for b directly.
- Apply same solution to both sides:

$$
\begin{aligned}
0 &= f\left(x_t, b'\psi(x_t, z_t), z_t, \mathcal{E}_t\right) \\
\mathcal{E}_t &= E_t\left[q\left(x_{t+1}, b'\psi(x_{t+1}, z_{t+1}), z_{t+1}\right)\right].
\end{aligned}
$$

- Run nonlinear equation solver with gradient

$$
\frac{\partial f}{\partial b} = \frac{\partial f}{\partial y_t}\frac{\partial y_t}{\partial b} + \frac{\partial f}{\partial \mathcal{E}_t} E_t\left[\left(\frac{\partial q}{\partial x_{t+1}} + \frac{\partial q}{\partial y_{t+1}}\frac{\partial y_{t+1}}{\partial x_{t+1}}\right)\frac{\partial x_{t+1}}{\partial y_t}\frac{\partial y_t}{\partial b} + \frac{\partial q}{\partial y_{t+1}}\frac{\partial y_{t+1}}{\partial b}\right]
$$

- Can start with time iteration and then switch to direct solution to get best of both worlds.

Special Case: Exogenous Asset Pricing

- Return to special case

$$PD(z_t) = E_t \left\{ M(z_t, z_{t+1}) \left(PD(z_{t+1}) + 1 \right) \frac{D(z_{t+1})}{D(z_t)} \right\}.$$

- Apply guess $PD(z_t) = \psi(z_t)'b$, and use quadrature scheme $(\omega_j, \bar{\varepsilon}_j)$:

$$\psi(z_t)'b = \sum_j \omega_j M(z_t, \bar{\varepsilon}_j) \left(\frac{D(z_t, \bar{\varepsilon}_j)}{D(z_t)} \right) \left(\psi(z_t, \bar{\varepsilon}_j)'b + 1 \right)$$

  with slight abuse of notation to substitute $(z_t, \bar{\varepsilon}_j)$ for z_{t+1}.

Special Case: Exogenous Asset Pricing

- If we now define

$$A(z_t) = \sum_j \omega_j M(z_t, \bar{\varepsilon}_j) \left(\frac{D(z_t, \bar{\varepsilon}_j)}{D(z_t)} \right) \psi(z_t, \bar{\varepsilon}_j)$$

$$c(z_t) = \sum_j \omega_j M(z_t, \bar{\varepsilon}_j) \left(\frac{D(z_t, \bar{\varepsilon}_j)}{D(z_t)} \right)$$

then we obtain

$$\Psi b = \mathbf{A} b + \mathbf{c}$$

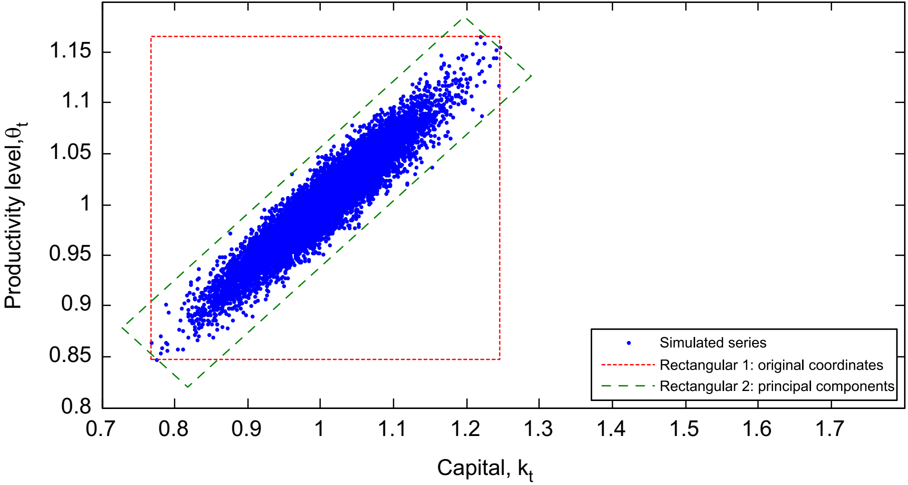
$$b = (\Psi - \mathbf{A})^{-1} \mathbf{c}$$

- Another one-step linear solution!

Additional Refinements

- **Precomputation**:
  - Can save time by pre-computing $\psi(\bar{x}_i, \bar{z}_i)$.
  - Note that for a given grid $\{\bar{z}_i\}$ and nodes $\{\bar{\varepsilon}_j\}$, end up with the same grid over z_{t+1}.
  - If $\psi(x_{t+1}, z_{t+1}) = \psi_x(x_{t+1})\psi_z(z_{t+1})$, then we can also precompute $\psi_z(z_{t+1}(\bar{z}_i, \bar{\varepsilon}_j))$.
- **Adaptive domain**:
  - Approximations work better when variables are not highly correlated.
  - Better: use principal components as states.
  - Do SVD to obtain $X = (x_1, \ldots, x_T)' = USV'$, then the PCs are $\tilde{X} = XV$.
  - Recover X using $X = \tilde{X}V'$.

Adaptive Domain: Illustration



Source: Judd, Maliar, Maliar, Valero (2013).

Special Case: Endogenous Grid Method

- Typically choose grid, then solve so optimality conditions hold on it.
  - But sometimes can skip optimization step by exploiting properties of equilibrium condition.
  - Endogenous grid method of Carroll (2006).
- Example: consider a life-cycle Bewley model with Euler equation

$$c_t(a_t, y_t)^{-\gamma} = \beta E_t \left[c_{t+1}(a_{t+1}, y_{t+1})^{-\gamma} \right]$$

where t is age, a is assets, and y is income, subject to the budget constraint

$$c_t + R^{-1} a_{t+1} = a_t + y_t$$

- Given (a_t, y_t), we cannot solve for c_t (equivalently, a_{t+1}) in closed form.

Special Case: Endogenous Grid Method

- But what if we somehow knew next period's value of a_{t+1} and next period's policy function c_{t+1}? Then from the Euler equation we would know

$$c_t^* = \left\{ \beta E_t \left[c_{t+1}(a_{t+1}, y_{t+1})^{-\gamma} \right] \right\}^{-1/\gamma}$$

  and from the budget constraint we would know

$$a_t^* = c_t^* + R^{-1} a_{t+1} - y_t.$$

- This means that if we start at (a_t^*, y_t), c_t^* is the optimal policy!
- For a given grid of a_{t+1} values, we can solve for (c_t^*, a_t^*, y_t), then approximate $c_t(a_t, y_t)$ on this grid (typically by linearly interpolating).
- Not a generic method, but when it works it is very simple and effective.
  - See Maliar and Maliar (2013), and other work for similar envelope condition method.

Perturbation Methods

Perturbation Methods

- Based on **local** expansion around a point.
- Computationally cheap, but less accurate far from approximation point.
  - Great as initial guess for global solution.
- Fold exogenous states into x_t to rewrite

$$
\begin{aligned}
0 &= E_t\left[f(x_t, y_t, x_{t+1}, y_{t+1})\right] \\
y_t &= g(x_t, \sigma) \\
x_{t+1} &= h(x_t, \sigma) + \sigma \eta \varepsilon_{t+1}.
\end{aligned}
$$

- Note that g and h are different from earlier notation.

Perturbation Methods

- First order perturbation:

$$y_t = g(\bar{x}, 0) + g_x(\bar{x}, 0)(x - \bar{x}) + g_\sigma(\bar{x}, 0)\sigma$$
$$x_{t+1} = h(\bar{x}, 0) + h_x(\bar{x}, 0)(x - \bar{x}) + h_\sigma(\bar{x}, 0)\sigma + \sigma\eta\varepsilon_{t+1}$$

- Choose values to set derivatives of equilibrium condition to zero:

$$(x_t): \quad 0 = \mathbf{f_x} + \mathbf{f_y g_x} + \mathbf{f_{x'} h_x} + \mathbf{f_{y'} g_x h_x} \tag{1}$$
$$(\sigma): \quad 0 = \mathbf{f_y g}_\sigma + \mathbf{f_{x'} h}_\sigma + \mathbf{f_{y'}}\left(\mathbf{g}_\sigma + \mathbf{g_x h}_\sigma\right) \tag{2}$$

- Solution to (2) implies $g_\sigma = h_\sigma = 0$ (no risk effects).

- Apply $g(\bar{x}, 0) = \bar{y}$, $h(\bar{x}, 0) = \bar{x}$, define e.g., $\hat{x} = x - \bar{x}$, to obtain system that must solve (1):

$$\hat{y}_t = g_x \hat{x}_t$$
$$\hat{x}_{t+1} = h_x \hat{x}_t + \sigma\eta\varepsilon_{t+1}$$

- Many solution techniques: Sims (2001), Klein (2000).

Higher Order Perturbations

- Second-order perturbation (see e.g., Judd and Guu (1997) for solution method):

$$\hat{y}_t = \mathbf{g_x}\hat{x}_t + \frac{1}{2}\mathbf{G}_{xx}(\hat{x}_t \otimes \hat{x}_t) + \frac{1}{2}\mathbf{g}_{\sigma\sigma}\sigma^2$$
$$\hat{x}_{t+1} = \mathbf{h_x}\hat{x}_t + \frac{1}{2}\mathbf{H}_{xx}(\hat{x}_t \otimes \hat{x}_t) + \frac{1}{2}\mathbf{h}_{\sigma\sigma}\sigma^2 + \sigma\eta\varepsilon_{t+1}$$

- Now risk influences policy functions (in a constant way).
  - Third-order: $\sigma^2\hat{x}_t$ term linear in states.
  - Higher order: nonlinear risk-state interactions.

- Major problem: explosiveness. Univariate example:

$$\hat{x}_{t+1} = \cdots + \frac{1}{2}h_{xx}\hat{x}_t^2 = \cdots + \frac{1}{2}h_{xx}\left(\cdots + \frac{1}{2}h_{xx}\hat{x}_{t-1}^2\right)^2$$

Pruned State Space

- **Pruned state-space** approach (Andreasen et al, 2018). Split x_t into first-order terms x_t^f and second-order terms x_t^s:

$$\hat{x}_{t+1}^f = \mathbf{h_x} \hat{x}_t^f + \sigma \eta \varepsilon_{t+1}$$
$$\hat{x}_{t+1}^s = \mathbf{h_x} \hat{x}_t^s + \frac{1}{2} \mathbf{H}_{xx} (\hat{x}_t^f \otimes \hat{x}_t^f) + \frac{1}{2} \mathbf{h}_{\sigma\sigma} \sigma^2$$

- No interaction between x^s and x^s means no explosiveness: $\hat{x}_t = A(L)\varepsilon_t + B(L)\varepsilon_t^2$.

- Policy functions:

$$\hat{y}_t = \mathbf{g_x} \left(\hat{x}_t^f + \hat{x}_t^s \right) + \frac{1}{2} \mathbf{G}_{xx} (\hat{x}_t^f \otimes \hat{x}_t^f) + \frac{1}{2} \mathbf{g}_{\sigma\sigma} \sigma^2$$

- See paper for third-order equivalent.

Perfect Foresight Paths

Perfect Foresight Paths

- Perfect foresight paths (also known as deterministic transition paths, or "MIT shocks")
- Idea: if we assume no risk from today on, then path back to steady state is solution to equilibrium conditions.
- Notation for equilibrium conditions:

$$f(s_{t-1}, s_t, s_{t+1}; z_t) = 0$$

  where $s' = (x', y')$ are endogenous states and policy functions, and z are exogenous states.

- Deterministic environment buys a lot of tractability.
  - Because no shocks will arrive, we can directly use s_{t+1}. Don't need expectations.
  - Can directly use nonlinear equilibrium conditions for f, no need to linearize.
  - Can change parameters or apply shocks to exogenous states.

Perfect Foresight Paths

- Solution and notation follow Juillard, Laxton, McAdam, Pioro (1998).
- Stack equations to form

$$\mathbf{f}(\mathbf{s}) = \begin{bmatrix} f_0(s_0) \\ f_1(s_0, s_1, s_2) \\ \vdots \\ f_T(s_{T-1}, s_T, s_{T+1}) \\ f_{T+1}(s_{T+1}) \end{bmatrix} = 0 \tag{3}$$

including additional initial and terminal equations, typically

$$f_0(s_0) = s_0 - s_0^* \qquad f_{T+1}(s_{T+1}) = s_{T+1} - s_{T+1}^*$$

where s_0^* is the initial steady state, and s_{T+1}^* is the final steady state.

Perfect Foresight Paths

- Typically solved using Newton's method:

$$\underbrace{\Delta \mathbf{s}}_{\text{step size}} = -\mathbf{f}'(\mathbf{s})^{-1}\mathbf{f}$$

- Key to solution is computing inverse Jacobian $\mathbf{f}'(\mathbf{s})^{-1}$. Want to solve:

$$\begin{bmatrix} I & & & & \\ L_1 & C_1 & F_1 & & \\ & \ddots & \ddots & \ddots & \\ & & L_T & C_T & F_T \\ & & & & I \end{bmatrix} \Delta \mathbf{s} = -\mathbf{f}$$

where L_t, C_t, and F_t are the derivatives of $f_t(s_{t-1}, s_t, s_{t+1})$ with respect to s_{t-1}, s_t, s_{t+1}.

Perfect Foresight Paths

- In practice, this matrix has size $n(T+2) \times n(T+2)$, where n is the number of equilibrium conditions and T is the number of periods.
  - Most of the entries are zeros, so sparse matrix tools can handle it.
  - Alternative: Juillard, Laxton, McAdam, Pioro (1998) provide a recursive algorithm computing $\Delta\mathbf{s}$.
- Weakness of the approach: exactly end at steady state.
  - May require huge number of periods to avoid distorting the calculations.
- My alternative: assume that by end of the sample equilibrium follows **linearized solution**.
  - Linearized solution: $y_t = G_x x_t + G_z z_t$ where y_t are endog. controls and x_t are endog. states.
  - Replace terminal condition with the following (h is transition equation):

$$f_{T+1}(s_t) = \begin{bmatrix} x_{T+1} - h(s_T, z_{T+1}) \\ y_{T+1} - G_x x_{T+1} - G_z z_{T+1} \end{bmatrix} = 0.$$

Sequence Space Jacobian

Sequence Space Jacobian

- Some questions require heterogeneous agent models.
  - Although you should keep in mind that some do not.
- In these cases, working with endogenous aggregate states is complex.
  - Often, only a small subset of aggregate quantities (e.g., prices) matter for individual behavior.
  - However, values of these aggregates may depend on the entire distribution.
  - Krusell-Smith approach approximates using simpler forecasts based only on moments.
  - But computationally intensive, and no guarantee this will work well.
- Recent alternative: **sequence space jacobian**.
  - Method to compute linearized impulse responses or perfect foresight paths.
  - Note: these solutions remove aggregate risk, but not idiosyncratic risk.

Sequence Space Jacobian

- Start with a function that defines the aggregate equilibrium $H = 0$.
- Example in neoclassical production model, capital market clearing:

$$H_t(K, Z) = \int_i k_{i,t}(X, Z) - K_t.$$

- For linearized impulse response, can use approximation

$$H_K dK + H_Z dZ = 0$$

  to obtain

$$dK = -H_K^{-1} H_Z dZ$$

- Can also solve this $H(K, Z) = 0$ as nonlinear system of equations.
- Key to both solutions is the **Jacobian**, (H_K, H_Z).

Sequence Space Jacobian

- First, we need to split the problem.
  - Het. agent models generally intractable when behavior depends on entire distribution.
  - Need to collapse to a subset of aggregate states X_t sufficient for the individual's problem.
  - In classical Krusell-Smith model, this is just prices (r_t, w_t).
- Define **block** $Y = h(X)$ to be mapping between sufficient states X and aggregate outputs Y.
  - In this example, X is prices, Y is capital demand.
  - Full model equilibrium requires multiple blocks:

    $$H(K, Z) = H(h(X), Z) = H(h(g(K, Z))Z)$$

    where $g(K, Z)$ maps states into prices (equal marginal products from firm FOCs).
  - Efficiency gains from applying closed form solutions when available (see paper).
- Then can compute Jacobian (H_K, H_Z) using the chain rule given Jacobians of h, $X(\cdot)$.

Sequence Space Jacobian

- Define notation for the problem as

$$\begin{aligned} \text{Individual optimality:} \quad & \mathbf{v}_t = v(\mathbf{v}_{t+1}, \mathbf{X}_t) \\ \text{Distribution law of motion:} \quad & \mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, \mathbf{X}_t)' \mathbf{D}_t \\ \text{Measurement of agg. states:} \quad & \mathbf{Y}_t = y(\mathbf{v}_{t+1}, \mathbf{X}_t)' \mathbf{D}_t. \end{aligned}$$

- Apply a single shock of size dx to X at time s.
  - Then we want to compute dY_t^s, change in Y at time t due to shock at time s.
- Take limit as $dx \to 0$: $\quad dY_t^s = (d\mathbf{y}_t^s)' \mathbf{D}_t^s + (\mathbf{y}_t^s)' d\mathbf{D}_t^s$
- Possible (but costly) to compute directly.
  - Apply the shock at time s, solve **v** backwards, then iterate **D** forwards.
  - Repeat this for each time s.

Sequence Space Jacobian

- First efficiency gain: policy functions depend only on distance to shock $s - t$

$$\mathbf{y}_t^s = \mathbf{y}_{t+k}^{s+k}, \qquad \Lambda_t^s = \Lambda_{t+k}^{s+k}.$$

- Second gain: use the fact that $dx \to 0$ to simplify the problem

$$dY_t^s = (d\mathbf{y}_t^s)'\mathbf{D}_t^s + (\mathbf{y}_t^s)'d\mathbf{D}_t^s = (d\mathbf{y}_t^s)'\mathbf{D}_{ss} + \mathbf{y}_{ss}'d\mathbf{D}_t^s.$$

- Now subtract dY_{t-1}^{s-1}:

$$\begin{aligned} dY_t^s - dY_{t-1}^{s-1} &= \underbrace{(d\mathbf{y}_t^s - d\mathbf{y}_{t-1}^{s-1})'}_{=0}\mathbf{D}_{ss} + \mathbf{y}_{ss}'(d\mathbf{D}_t^s - d\mathbf{D}_{t-1}^{s-1}) \\ &= \mathbf{y}_{ss}'(d\mathbf{D}_t^s - d\mathbf{D}_{t-1}^{s-1}) \end{aligned}$$

Sequence Space Jacobian

- Difference in distributions:

$$d\mathbf{D}_t^s = (d\Lambda_{t-1}^s)'\mathbf{D}_{ss} + (\Lambda_{ss})'d\mathbf{D}_{t-1}$$

- Now difference as in previous slide:

$$\begin{aligned} d\mathbf{D}_t^s - d\mathbf{D}_{t-1}^{s-1} &= \underbrace{(d\Lambda_{t-1}^s - d\Lambda_{t-2}^{s-1})'}_{=0}\mathbf{D}_{ss} + \Lambda_{ss}'(d\mathbf{D}_{t-1} - d\mathbf{D}_{t-2}) \\ &= \Lambda_{ss}'(d\mathbf{D}_{t-1} - d\mathbf{D}_{t-2}) \\ &\vdots \\ &= (\Lambda_{ss}')^{t-1}(d\mathbf{D}_1 - d\mathbf{D}_0) \\ &= (\Lambda_{ss}')^{t-1}(d\Lambda_0^1)'\mathbf{D}_{ss} \end{aligned}$$

- Recursive structure re-using repeated terms much cheaper to compute.

Conclusion

- Many tools available, want to select right tools for the right job.
- More complex or high tech is not always better!
  - Simpler models are often easier to understand.
  - You can run lots of things to gain intuition about role of different mechanisms.
  - You retain degrees of freedom to use on other features.
- My advice: start with simple methods before complexifying.
  - My personal favorite: perfect foresight paths.