Complementary Slides to "A Machine Learning Projection Method for Macro-Finance Models"

Vytautas Valaitis and Alessandro T. Villa

February 22, 2023

Model

Households:

$$\max_{c_t, k_{t+1}} E_0 \sum_{t=0}^{\infty} \beta^t U(c) \quad s.t.$$

$$k_t(1 + r_t - \delta) + w_t n_t \geq c_t + k_{t+1}$$

Firms:

$$\max_{K_t, N_t} z_t F(K_t, N_t) - w_t N_t - r_t K_t$$

Technology:

$$ln(z_t) = \rho ln(z_{t-1}) + \epsilon_t \quad \epsilon_t \sim N(0, \sigma)$$

Equilibrium:

$$k_t = K_t; \quad n_t = N_t$$

$$r_t = z_t F_K(K_t, N_t); \quad w_t = z_t F_N(K_t, N_t)$$

Back

Main approaches to solving macroeconomic models

Main approaches to solving economic models:

- **Local**
  - Perturbation.
- **Global**
  - Grid-based: Value function iteration, policy function iteration, endogenous gridpoint method, projection.
  - Grid-free: Parameterized expectations.

Recursive Representation

$$V(k,z) = \max_{c,k'} U(c) + \beta E[V(k',z')|z]$$

$$k(1 + r(z,k) - \delta) + w(z,k) - k' = c$$

Note that at this point I impose the equilibrium: $N_t = 1$, $r(z,k) = F_K(z,K,N)$, $w(z,k) = F_K(z,K,N)$.

The purpose is to find:

- Value function: $V(k,z)$.
- Policy functions: $k'(k,z), c(k,z)$.

Algorithm

Define the grid for k and z and guess the value function $V^i(k,z)$. Denote $g^i(k,z)$ an optimal choice of k' given the value function $V^i(k,z)$

1. Given a guess $V^i(k,z)$ solve for $g^i(k,z)$

$$g^i(k,z) = argmax \quad U(k, g^i(k,z)) + \beta\, \mathbb{E}[V^i(g^i(k,z), z')|z]$$

2. Update the value function

$$V^{i+1}(k,z) = U(k, g^i(k,z)) + \beta\, \mathbb{E}[V^i(g^i(k,z), z')|z]$$

Algorithm

Define the grid for k and z and guess the value function $V^i(k,z)$. Denote $g^i(k,z)$ an optimal choice of k' given the value function $V^i(k,z)$

1. Given a guess $V^i(k,z)$ solve for $g^i(k,z)$

$$g^i(k,z) = argmax \quad U(k, g^i(k,z)) + \beta\, \mathbb{E}[V^i(g^i(k,z), z')|z]$$

2. Update the value function

$$V^{i+1}(k,z) = U(k, g^i(k,z)) + \beta\, \mathbb{E}[V^i(g^i(k,z), z')|z]$$

3. Update the guess $V^{i+1}(k,z) = V^i(k,z)$. Check convergence, i.e. $error = ||V^{i+1}(k,z) - V^i(k,z)||$. If convergence criterion not satisfied, go back to (1)

Algorithm

Define the grid for k and z and guess the value function $V^i(k,z)$. Denote $g^i(k,z)$ an optimal choice of k' given the value function $V^i(k,z)$

1. Given a guess $V^i(k,z)$ solve for $g^i(k,z)$

$$g^i(k,z) = argmax \quad U(k, g^i(k,z)) + \beta\, \mathbb{E}[V^i(g^i(k,z), z')|z]$$

2. Update the value function

$$V^{i+1}(k,z) = U(k, g^i(k,z)) + \beta\, \mathbb{E}[V^i(g^i(k,z), z')|z]$$

3. Update the guess $V^{i+1}(k,z) = V^i(k,z)$. Check convergence, i.e. $error = ||V^{i+1}(k,z) - V^i(k,z)||$. If convergence criterion not satisfied, go back to (1)

Once done, can back out $c(k,z)$ from the budget constraint.

Projection

Recall the optimality conditions, denote the euler equation error by $e(k, z)$.

$$e(k, z) = u'(c) - \beta \, \mathbb{E}[u'(c')(r(k', z') + 1 - \delta)|z]$$

Projection

Recall the optimality conditions, denote the euler equation error by $e(k,z)$.

$$e(k,z) = u'(c) - \beta\,\mathbb{E}[u'(c')(r(k',z') + 1 - \delta)|z]$$

If policy functions satisfy rational expectations equilibrium, $e(k,z) = 0$.
You can parametrize the policy $c(k,z) = P(k,z,\eta_n)$ and find coefficients (η_n) such that $e(k,z)$ is minimized.

Projection

Recall the optimality conditions, denote the euler equation error by $e(k,z)$.

$$e(k,z) = u'(c) - \beta\,\mathbb{E}[u'(c')(r(k',z') + 1 - \delta)|z]$$

If policy functions satisfy rational expectations equilibrium, $e(k,z) = 0$.
You can parametrize the policy $c(k,z) = P(k,z,\eta_n)$ and find coefficients (η_n) such that $e(k,z)$ is minimized.

$P(k_t, z_t, \eta_n)$ - an approximating function, typically a polynomial or spline.

Projection

Recall the optimality conditions, denote the euler equation error by $e(k,z)$.

$$e(k,z) = u'(c) - \beta\,\mathbb{E}[u'(c')(r(k',z') + 1 - \delta)|z]$$

If policy functions satisfy rational expectations equilibrium, $e(k,z) = 0$.
You can parametrize the policy $c(k,z) = P(k,z,\eta_n)$ and find coefficients (η_n) such that $e(k,z)$ is minimized.

$P(k_t, z_t, \eta_n)$ - an approximating function, typically a polynomial or spline.

Instead of solving for $V(k,z)$ (infinite dimensional), need to solve for η_n (finite dimensional).

Projection

M grid points gives M equations to solve for N coefficients. Need to have at least $M \geq N$. (more grid points than parameters to solve for).

$$e(k_i, z_i) = u'(P(k_i, z_i, \eta_n)) - \mathbb{E}[u'(P(k_i', z_i', \eta_n))(z' k'^{\alpha-1}\alpha + 1 - \delta)|z] \tag{1}$$

Express (1) in a way that the only unknown is η_n.

$$e(k_i, z_i) = u'(P(k_i, z_i, \eta_n)) - \sum_{i=1}^{J}[w_i u'(P(k_i', x_i', \eta_n))(\alpha z' k'^{\alpha-1} + 1 - \delta)] \tag{2}$$

and given $P(k_i, z_i, \eta_n)$, k' is known from the budget constraint:

$$P(k_i, z_i, \eta_n) + k_i' = z k_i^{\alpha} + (1 - \delta)k_i$$

We parameterized $c(k, z) = P(k, z, \eta_n)$, can we, instead, parameterize $k'(k, z)$.

Projection

Different ways to solve for η_n:

Projection

Different ways to solve for η_n:

- Solver: when $M = N$ can solve exactly for $e(k_i, z_i) = 0 \forall i$, when M>N need to do some weighting.

Projection

Different ways to solve for η_n:

- Solver: when $M = N$ can solve exactly for $e(k_i, z_i) = 0 \forall i$, when M>N need to do some weighting.
- Iteration:

Projection

Different ways to solve for η_n:

- Solver: when $M = N$ can solve exactly for $e(k_i, z_i) = 0 \forall i$, when M>N need to do some weighting.
- Iteration:
  - Time iteration.

Projection

Different ways to solve for η_n:

- Solver: when $M = N$ can solve exactly for $e(k_i, z_i) = 0 \forall i$, when M>N need to do some weighting.
- Iteration:
  - Time iteration.
  - Fixed point iteration.

Projection

Different ways to solve for η_n:

- Solver: when $M = N$ can solve exactly for $e(k_i, z_i) = 0 \forall i$, when M>N need to do some weighting.
- Iteration:
  - Time iteration.
  - Fixed point iteration.

Discussion:

- Solver: more efficient updating of η_n, but increasingly unreliable when N is large.

Projection

Different ways to solve for η_n:

- Solver: when $M = N$ can solve exactly for $e(k_i, z_i) = 0 \forall i$, when M>N need to do some weighting.
- Iteration:
  - Time iteration.
  - Fixed point iteration.

Discussion:

- Solver: more efficient updating of η_n, but increasingly unreliable when N is large.
- Iteration: better convergence properties, can handle large N, but may require a large number of iterations iterations.

Projection: Solver

1 Create grids for k, z.

Projection: Solver

1 Create grids for k, z.

2 For each i on the grid, define $e(k_i, z_i)$

$$e(k_i, z_i) = u'(P(k_i, z_i, \eta_n)) - \mathbb{E}[u'(P(k_i', z_i', \eta_n))(r(k', z') + 1 - \delta)|z]$$

where can use quadrature to calculate $\mathbb{E}$.
get k' from the resource constraint: $k_i' = zk_i^\alpha + (1-\delta)k_i - P(k_i, z_i, \eta_n)$.

Projection: Solver

1 Create grids for k, z.

2 For each i on the grid, define $e(k_i, z_i)$

$$e(k_i, z_i) = u'(P(k_i, z_i, \eta_n)) - \mathbb{E}[u'(P(k_i', z_i', \eta_n))(r(k', z') + 1 - \delta)|z]$$

where can use quadrature to calculate $\mathbb{E}$.
get k' from the resource constraint: $k_i' = zk_i^\alpha + (1-\delta)k_i - P(k_i, z_i, \eta_n)$.

3 Use a solver of your choice to minimize $e(k, z)$.

Projection: Solver

1. Create grids for k, z.
2. For each i on the grid, define $e(k_i, z_i)$

$$e(k_i, z_i) = u'(P(k_i, z_i, \eta_n)) - \mathbb{E}[u'(P(k_i', z_i', \eta_n))(r(k', z') + 1 - \delta)|z]$$

   where can use quadrature to calculate $\mathbb{E}$.
   get k' from the resource constraint: $k_i' = zk_i^\alpha + (1-\delta)k_i - P(k_i, z_i, \eta_n)$.

3. Use a solver of your choice to minimize $e(k, z)$.

$\rightarrow$ everything done in one shot!

Projection: Iteration

Projection: Iteration

1 Create grids for k, z.

Projection: Iteration

1 Create grids for k, z.

2 For each i on the grid, calculate c_i.

$$c_i = u'^{-1}\left(\mathbb{E}[u'(P(k_i', z_i', \eta_n^{q-1}))(r(k', z') + 1 - \delta)|z]\right)$$

where can use quadrature to calculate $\mathbb{E}$.
get k' from the resource constraint: $k_i' = r(k_i, z_i) + (1-\delta)k_i - P(k_i, z_i, \eta_n^q)$.

Projection: Iteration

1 Create grids for k, z.

2 For each i on the grid, calculate c_i.

$$c_i = u'^{-1}\left(\mathbb{E}[u'(P(k_i', z_i', \eta_n^{q-1}))(r(k', z') + 1 - \delta)|z]\right)$$

where can use quadrature to calculate $\mathbb{E}$.
get k' from the resource constraint: $k_i' = r(k_i, z_i) + (1-\delta)k_i - P(k_i, z_i, \eta_n^q)$.

3 Update η_n by projecting c_i on k_i, z_i.

$$\eta_n = argmin \sum_{i=1}^{M} (c_i - P(k_i, z_i))^2$$

Projection: Iteration

1 Create grids for k, z.

2 For each i on the grid, calculate c_i.

$$c_i = u'^{-1}\left(\mathbb{E}[u'(P(k_i', z_i', \eta_n^{q-1}))(r(k', z') + 1 - \delta)|z]\right)$$

where can use quadrature to calculate $\mathbb{E}$.
get k' from the resource constraint: $k_i' = r(k_i, z_i) + (1-\delta)k_i - P(k_i, z_i, \eta_n^q)$.

3 Update η_n by projecting c_i on k_i, z_i.

$$\eta_n = argmin \sum_{i=1}^{M} (c_i - P(k_i, z_i))^2$$

4 Do some dampening: $\eta_n^q = \omega\eta_n + (1-\omega)\eta_n^q$.

Projection: Iteration

1 Create grids for k, z.

2 For each i on the grid, calculate c_i.

$$c_i = u'^{-1}\left(\mathbb{E}[u'(P(k_i', z_i', \eta_n^{q-1}))(r(k', z') + 1 - \delta)|z]\right)$$

where can use quadrature to calculate $\mathbb{E}$.
get k' from the resource constraint: $k_i' = r(k_i, z_i) + (1-\delta)k_i - P(k_i, z_i, \eta_n^q)$.

3 Update η_n by projecting c_i on k_i, z_i.

$$\eta_n = argmin \sum_{i=1}^{M} (c_i - P(k_i, z_i))^2$$

4 Do some dampening: $\eta_n^q = \omega\eta_n + (1-\omega)\eta_n^q$.

5 Repeat until $\eta_n^q = \eta_n^{q-1}$.

Projection: Iteration

1 Create grids for k, z.

2 For each i on the grid, calculate c_i.

$$c_i = u'^{-1}\left(\mathbb{E}[u'(P(k_i', z_i', \eta_n^{q-1}))(r(k', z') + 1 - \delta)|z]\right)$$

where can use quadrature to calculate $\mathbb{E}$.
get k' from the resource constraint: $k_i' = r(k_i, z_i) + (1-\delta)k_i - P(k_i, z_i, \eta_n^q)$.

3 Update η_n by projecting c_i on k_i, z_i.

$$\eta_n = argmin \sum_{i=1}^{M} (c_i - P(k_i, z_i))^2$$

4 Do some dampening: $\eta_n^q = \omega\eta_n + (1-\omega)\eta_n^q$.

5 Repeat until $\eta_n^q = \eta_n^{q-1}$.

Dampening - inevitable element of numerical work.

Stochastic Simulations PEA

Idea: use the optimality conditions and parameterize the expected value terms with some parametric functions. See Den Haan & Marcet 1990 for reference.

1. Assume that policy for c_t follows some parameterized form $P(k_t, z_t, \eta_n^q)$.
2. Generate $\{c_t, k_{t+1}\}_{t=1}^{T}$ using the optimality conditions.

$$c_t = P(k_t, z_t, \eta_n^q)$$
$$k_{t+1} = z_t k_t^{\alpha} + (1-\delta)k_t - c_t$$

Stochastic Simulations PEA

Idea: use the optimality conditions and parameterize the expected value terms with some parametric functions. See Den Haan & Marcet 1990 for reference.

1. Assume that policy for c_t follows some parameterized form $P(k_t, z_t, \eta_n^q)$.
2. Generate $\{c_t, k_{t+1}\}_{t=1}^T$ using the optimality conditions.

$$c_t = P(k_t, z_t, \eta_n^q)$$
$$k_{t+1} = z_t k_t^\alpha + (1-\delta)k_t - c_t$$

3. Generate the $Y \equiv \mathbb{E}[]$ term using $\{c_t, k_t, z_t\}_{t=1}^T$

$$Y_t = u'(c_{t+1})(\alpha z_{t+1} k_{t+1}^{\alpha-1} + (1-\delta))$$

Stochastic Simulations PEA

Idea: use the optimality conditions and parameterize the expected value terms with some parametric functions. See Den Haan & Marcet 1990 for reference.

1. Assume that policy for c_t follows some parameterized form $P(k_t, z_t, \eta_n^q)$.
2. Generate $\{c_t, k_{t+1}\}_{t=1}^T$ using the optimality conditions.

$$c_t = P(k_t, z_t, \eta_n^q)$$
$$k_{t+1} = z_t k_t^\alpha + (1-\delta)k_t - c_t$$

3. Generate the $Y \equiv \mathbb{E}[]$ term using $\{c_t, k_t, z_t\}_{t=1}^T$

$$Y_t = u'(c_{t+1})(\alpha z_{t+1} k_{t+1}^{\alpha-1} + (1-\delta))$$

4. Update coefficients using the simulated $\{c_t, k_t, z_t\}_{t=1}^T$

$$\eta_n = \arg\min_{\eta_n} \frac{1}{T} \sum_{i=t^{\text{start}}}^{T} (Y_t - P(k_t, z_t, \eta_n))^2$$

Stochastic Simulations PEA

Idea: use the optimality conditions and parameterize the expected value terms with some parametric functions. See Den Haan & Marcet 1990 for reference.

1. Assume that policy for c_t follows some parameterized form $P(k_t, z_t, \eta_n^q)$.
2. Generate $\{c_t, k_{t+1}\}_{t=1}^T$ using the optimality conditions.

$$c_t = P(k_t, z_t, \eta_n^q)$$
$$k_{t+1} = z_t k_t^\alpha + (1-\delta)k_t - c_t$$

3. Generate the $Y \equiv \mathbb{E}[]$ term using $\{c_t, k_t, z_t\}_{t=1}^T$

$$Y_t = u'(c_{t+1})(\alpha z_{t+1} k_{t+1}^{\alpha-1} + (1-\delta))$$

4. Update coefficients using the simulated $\{c_t, k_t, z_t\}_{t=1}^T$

$$\eta_n = \arg\min_{\eta_n} \frac{1}{T} \sum_{i=t^{\text{start}}}^{T} (Y_t - P(k_t, z_t, \eta_n))^2$$

5. Use dampening: $\eta_n^{q+1} = \omega\eta_n + (1-\omega)\eta_n^q$

Stochastic simulation PEA

Advantages:

- Grid-free, so no curve of dimensionality.
- Do not need to include irrelevant points of the state-space.

Stochastic simulation PEA

Advantages:

- Grid-free, so no curve of dimensionality.
- Do not need to include irrelevant points of the state-space.

Disadvantages:

- $X'X$ may be low rank (if variables are multi-collinear and/or use higher order polynomials in your)
- Sampling error disappears slowly.
- No convergence properties. So need a good initial guess.

How to Parameterize

- Polynomials: Naive:

$$P(k_t, z_t; \eta_n) = \eta_0 + \eta_1 k_t + \eta_2 z_t$$

$$P(k_t, z_t; \eta_n) = exp(log(\eta_0 + \eta_1 k_t + \eta_2 z_t))$$

Can add higher order terms, use orthogonal polynomials (Chebyshev), etc.

How to Parameterize

- Polynomials: Naive:

$$P(k_t, z_t; \eta_n) = \eta_0 + \eta_1 k_t + \eta_2 z_t$$

$$P(k_t, z_t; \eta_n) = exp(log(\eta_0 + \eta_1 k_t + \eta_2 z_t))$$

  Can add higher order terms, use orthogonal polynomials (Chebyshev), etc.

- Artificial Neural Network

Parameterized-Expectations and Neural Networks

What is an ANN?

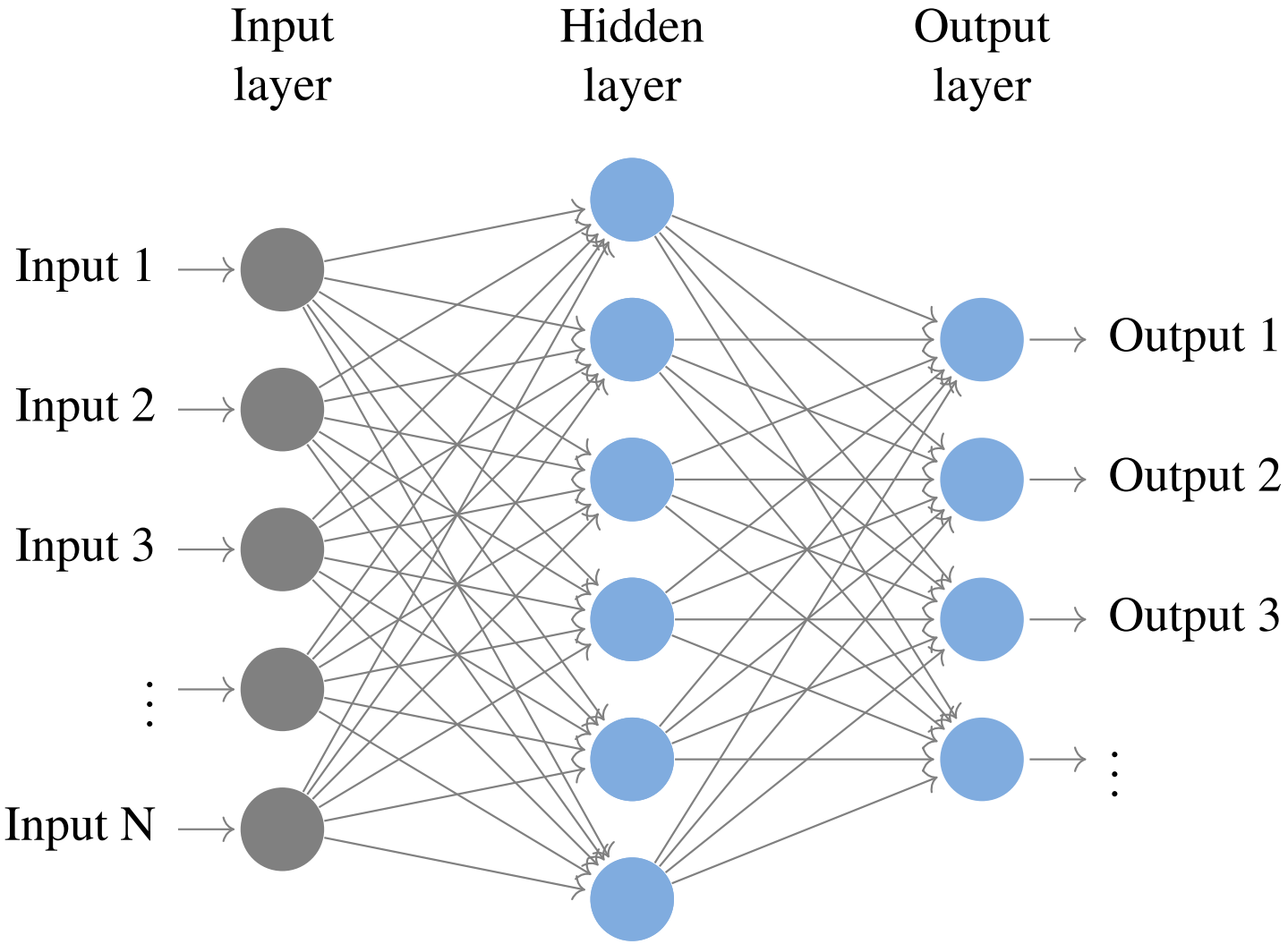


Figure: Artificial Neural Network Structure

Notes: The figure presents the structure of a single hidden layer artificial neural network. Each circle in t
neuron, and the arrows point the direction of the information flow in the prediction process. Neurons in the hi

What is ANN?

Network:

$$\tilde{X}_m = \mathcal{H}\left(\sum_{s=0}^{S} w_{m,s} \cdot X_{s,t}\right), \quad m = 1, \cdots, M,$$

$$\mathcal{F}_e(X_t; w, \psi) = \psi_{0,e} + \sum_{m=1}^{M} \psi_{m,e} \cdot \tilde{X}_m, \quad e = 1, \cdots, E.$$

Transfer function:

$$\mathcal{H}(x) = \frac{1}{1 + \exp(-x)}.$$

What is an ANN? Terminology

- **Hyper parameters**: Describe the ANN structure and how it is to be trained.

What is an ANN? Terminology

- **Hyper parameters**: Describe the ANN structure and how it is to be trained.
- **Weights**: parameters that combined inputs to produce outputs inside a neuron.

What is an ANN? Terminology

- **Hyper parameters**: Describe the ANN structure and how it is to be trained.
- **Weights**: parameters that combined inputs to produce outputs inside a neuron.
- **Back-Propagation**: method to efficiently calculate partial derivatives of the ANN weights.

What is an ANN? Terminology

- **Hyper parameters**: Describe the ANN structure and how it is to be trained.
- **Weights**: parameters that combined inputs to produce outputs inside a neuron.
- **Back-Propagation**: method to efficiently calculate partial derivatives of the ANN weights.
- **Neurons**: a mathematical operation that takes inputs and transforms them into outputs by applying weighting and passing them through the transfer function.

What is an ANN? Terminology

- **Hyper parameters**: Describe the ANN structure and how it is to be trained.
- **Weights**: parameters that combined inputs to produce outputs inside a neuron.
- **Back-Propagation**: method to efficiently calculate partial derivatives of the ANN weights.
- **Neurons**: a mathematical operation that takes inputs and transforms them into outputs by applying weighting and passing them through the transfer function.
- **Transfer Function**: function that converts neuron inputs into outputs.

What is an ANN? Terminology

- **Hyper parameters**: Describe the ANN structure and how it is to be trained.
- **Weights**: parameters that combined inputs to produce outputs inside a neuron.
- **Back-Propagation**: method to efficiently calculate partial derivatives of the ANN weights.
- **Neurons**: a mathematical operation that takes inputs and transforms them into outputs by applying weighting and passing them through the transfer function.
- **Transfer Function**: function that converts neuron inputs into outputs.
- **Epochs**: One epoch is when when a dataset is passed forward and backward through the neural network once. Number of epochs is the number of times the data is passed through the backward and forward stage.

What is an ANN? Terminology

- **Hyper parameters**: Describe the ANN structure and how it is to be trained.
- **Weights**: parameters that combined inputs to produce outputs inside a neuron.
- **Back-Propagation**: method to efficiently calculate partial derivatives of the ANN weights.
- **Neurons**: a mathematical operation that takes inputs and transforms them into outputs by applying weighting and passing them through the transfer function.
- **Transfer Function**: function that converts neuron inputs into outputs.
- **Epochs**: One epoch is when when a dataset is passed forward and backward through the neural network once. Number of epochs is the number of times the data is passed through the backward and forward stage.
- **Learning rate**: rate at which network weights adjust to prediction error during the training phase.

Number of Neurons?

Exploit the trade-off between the fit and prediction accuracy

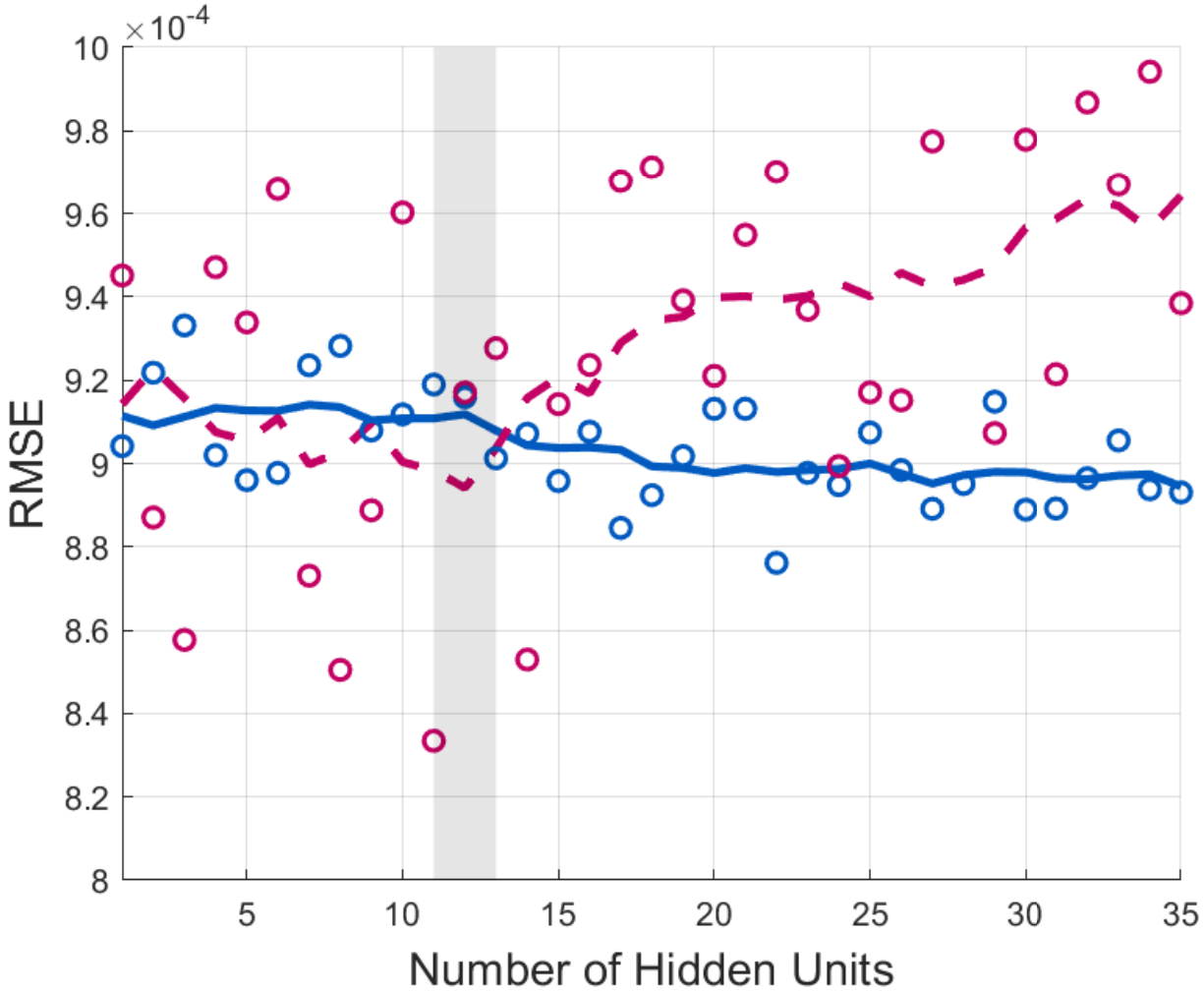


Figure: Source: Valaitis and Villa 2021. Blue: performance on the training set. Red: performance on the validation set.

Recursive Contracts (Marcet & Marimon 2019)

A village economy: risk-sharing

Setting:

- Moneylender: $\mathbb{E}_0 \beta^t c_t^l$
- Villager: $\mathbb{E}_0 \beta^t u(c_t)$
- Villager's stochastic income y_t Markov
- Resource constraint: $c_t + c_t^l = y_t$

Village economy: participation constraint

In order for the villager to stay in a risk-sharing contract, this must satisfy for all t,

$$\mathbb{E}_t \sum_{j=0}^{\infty} \beta^j u(c_{t+j}) \geq v^d(y_t) \equiv \mathbb{E}_t \sum_{j=0}^{\infty} \beta^j u(y_{t+j}) - c^d$$

Because this constraint involves future variables, standard recursive formulation (Bellman) does not hold! Time-inconsistency (Kydland and Prescott, 1977). However, we can still find an alternative recursive formulation to solve for the time-inconsistent optimal contract under full commitment.

Langrangean 1

Let ϕ be the Pareto-weight on the villager and α the multiplier on the participation constraint.

$$L \equiv \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t (y_t - c_t + \phi u(c_t)) + \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \alpha_t \mathbb{E}_0 \sum_{j=0}^{\infty} (\beta^j u(c_{t+j}) - v^d(y_t))$$

Key step: collect the terms multiplying $u(c_t)$ in the participation constraints and apply the law of iterated expectations

$$\mathbb{E}_0 \beta^t u(c_t) \sum_{s=0}^{t} \alpha_s$$

and define $\mu_t = \sum_{s=0}^{t} \alpha_s = \mu_{t-1} + \alpha_t$, starting from $\mu\_1 = 0$.

Transformed problem

Lagrangean:

$$L = \mathbb{E}_0 \sum_{t=0}^{\infty} \left(y_t - c_t + u(c_t)(\phi + \mu_{t-1}) + \alpha_t (u(c_t) - v^d(y_t)) \right)$$

Transformed problem

Lagrangean:

$$L = \mathbb{E}_0 \sum_{t=0}^{\infty} \left(y_t - c_t + u(c_t)(\phi + \mu_{t-1}) + \alpha_t (u(c_t) - v^d(y_t)) \right)$$

"Recursive Lagrangean" or a saddle-point functional equation:

$$W(y, \mu) = \min_{\alpha > 0} \max_{c} y - c + u(c)(\phi + \mu) + \alpha(u(c) - v(y^d)) + \beta \mathbb{E} W(y', \mu')$$

given

$\mu_t = \mu_{t-1} + \alpha_t$

Standard value function iteration not directly applicable.

Optimality

The optimal contract satisfies

$$u'(c_t) = \frac{1}{\phi + \mu_t}$$

Notice that c_t follows a non-decreasing path. Every time the villager is tempted to default, he gets a permanent consumption increase. The permanent effects of these shocks are propagated through the co-state variable μ_t, where $\mu_t = \mu_{t-1} + \alpha_t$.

Solve using parameterized expectations

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ
2. Simulate a sequence of shocks y_t of length T. For each t:

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ
2. Simulate a sequence of shocks y_t of length T. For each t:
    1. Calculate c_t from the complementary slackness condition:

$$c_t = u^{-1}\left(v^d(y) - P(y_t, \mu_{t-1}; \phi)\right)$$

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ
2. Simulate a sequence of shocks y_t of length T. For each t:
    1. Calculate c_t from the complementary slackness condition:

$$c_t = u^{-1}\left(v^d(y) - P(y_t, \mu_{t-1}; \phi)\right)$$

    2. Calculate α_t using the first order condition: $\alpha_t = \frac{1}{u'(c_t)} - \Psi - \mu_{t-1}$.

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ
2. Simulate a sequence of shocks y_t of length T. For each t:
    1. Calculate c_t from the complementary slackness condition:

$$c_t = u^{-1}\left(v^d(y) - P(y_t, \mu_{t-1}; \phi)\right)$$

    2. Calculate α_t using the first order condition: $\alpha_t = \frac{1}{u'(c_t)} - \Psi - \mu_{t-1}$.
    3. if $\alpha_t \leq 0$, set $\alpha = 0$ and $c_t = u'^{-1}\left(\frac{1}{\Psi + \mu_{t-1}}\right)$.

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ
2. Simulate a sequence of shocks y_t of length T. For each t:
    1. Calculate c_t from the complementary slackness condition:

$$c_t = u^{-1}\left(v^d(y) - P(y_t, \mu_{t-1}; \phi)\right)$$

    2. Calculate α_t using the first order condition: $\alpha_t = \frac{1}{u'(c_t)} - \Psi - \mu_{t-1}$.
    3. if $\alpha_t \leq 0$, set $\alpha = 0$ and $c_t = u'^{-1}\left(\frac{1}{\Psi + \mu_{t-1}}\right)$.
    4. Update μ_t: $\mu_t = \mu_{t-1} + \alpha_t$.

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ
2. Simulate a sequence of shocks y_t of length T. For each t:
    1. Calculate c_t from the complementary slackness condition:

$$c_t = u^{-1}\left(v^d(y) - P(y_t, \mu_{t-1}; \phi)\right)$$

    2. Calculate α_t using the first order condition: $\alpha_t = \frac{1}{u'(c_t)} - \Psi - \mu_{t-1}$.
    3. if $\alpha_t \leq 0$, set $\alpha = 0$ and $c_t = u'^{-1}\left(\frac{1}{\Psi + \mu_{t-1}}\right)$.
    4. Update μ_t: $\mu_t = \mu_{t-1} + \alpha_t$.
3. Given the simulated sequence for c_t, calculate $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j})$ and update ϕ.

Solve using parameterized expectations

1. Parameterize $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j}) \approx P(y_t, \mu_{t-1}; \phi)$. Guess ϕ
2. Simulate a sequence of shocks y_t of length T. For each t:
    1. Calculate c_t from the complementary slackness condition:

$$c_t = u^{-1}\left(v^d(y) - P(y_t, \mu_{t-1}; \phi)\right)$$

    2. Calculate α_t using the first order condition: $\alpha_t = \frac{1}{u'(c_t)} - \Psi - \mu_{t-1}$.
    3. if $\alpha_t \leq 0$, set $\alpha = 0$ and $c_t = u'^{-1}\left(\frac{1}{\Psi + \mu_{t-1}}\right)$.
    4. Update μ_t: $\mu_t = \mu_{t-1} + \alpha_t$.
3. Given the simulated sequence for c_t, calculate $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j})$ and update ϕ.
4. Check if simulated $\mathbb{E}_t \sum_{j=1}^{\infty} \beta^j u(c_{t+j})$ and $P(y_t, \mu_{t-1}; \phi)$ are close enough and that ϕ are stable across iterations. If not, go back to step 2.

Algorithm: putting pieces together

How to calculate $v^d(y)$?

$$v^d(y_t) \equiv \mathbb{E}_t \sum_{j=0}^{\infty} \beta^j u(y_{t+j}) - c^d$$

c^d is a parameter.

The rest can be computed as a future discounted sum:

```
for t=1:chainLength
    futureDiscountedY(t)=beta.^(0:infinityHorizon-t)*u(y(
       t:infinityHorizon))';
end
```

Note that *infinityHorizon* >> *chainLength*

Applications of recursive contracts

A long list. For instance,

Dynamic principal-agent problems/financial contracts (Cooley, Marimon and Quadrini, 2004)

Growth without commitment (Marcet-Marimon, 1992)

International business cycles (Kehoe and Perri, 2002)

Optimal fiscal policy without state-contingent debt (Aiyagari et. al., 2002)

Incomplete Markets

We now abandon the assumption that financial markets are complete!

We no longer have state contingent debt. What types of debt make sense to consider?

Governments issue loads of short debt. We will first consider the case where debt is issued in one short term bond.

Incomplete Markets

We now abandon the assumption that financial markets are complete!

We no longer have state contingent debt. What types of debt make sense to consider?

Governments issue loads of short debt. We will first consider the case where debt is issued in one short term bond.

We will answer:

- How will tax rates behave under non-state contingent debt?
- How will debt behave through time?

Key reference is Aiyagari, Marcet, Sargent, and Seppalla (2002, AMSS).

Aiyagari et. al. 2002

Environment:

- Preferences: $\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t (u(c_t, l_t))$
- Technology: $c_t + g_t = n_t$.
- g_t is stochastic Markov process.
- The government can tax labor income and issue a single one-period non-contingent bond $b_{t+1}(s^t)$, subject to a maximum borrowing limit B.
- Government can use distortionary labor taxes (τ_t) and non-state contingent debt (b_{t+1}).

Competitive equilibrium

- Households:

$$c_t + q_t b_{t+1} = (1 - \tau_t) w_t n_t + b_t$$

  Optimality:

$$q_t(g^t) = \beta \mathbb{E}_t \frac{u_c(g^{t+1})}{u_c(g^t)} \qquad (1 - \tau_t) w_t = \frac{u_l(g^t)}{u_c(g^t)}$$

- Government:

$$b_{t+1} q_t + \tau_t w_t n_t = g_t + b_t$$

- Firms: $w_t = 1$

- Borrowing limits: $\underline{\mathrm{B}} \leq b_{t+1} \leq \bar{B}$

Complete vs Incomplete Markets

Denote the government surplus by $s_t \equiv \tau_t w_t n_t - g_t$.

$$b_t(g^{t-1}) = \mathbb{E}_t \sum_{j=0}^{\infty} \beta^t \frac{u_c(c_{t+j}(g^{t+j}))}{u_c(c_t(g^t))} s_{t+j}(g^{t+j}) \tag{3}$$

Under complete markets $b_t(g_t|g^{t-1})$ was measurable at g^t and was slack. Therefore the only relevant constraint was at t=0 because b_0 is fixed.

Complete vs Incomplete Markets

Denote the government surplus by $s_t \equiv \tau_t w_t n_t - g_t$.

$$b_t(g^{t-1}) = \mathbb{E}_t \sum_{j=0}^{\infty} \beta^t \frac{u_c(c_{t+j}(g^{t+j}))}{u_c(c_t(g^t))} s_{t+j}(g^{t+j}) \tag{3}$$

Under complete markets $b_t(g_t|g^{t-1})$ was measurable at g^t and was slack. Therefore the only relevant constraint was at t=0 because b_0 is fixed.

Under incomplete markets, the planner cannot buy state contingent insurance and therefore history matters. That is why the measurability constraint **??** needs to hold at every period and state.

Ramsey Problem 1

$$
\begin{aligned}
L = & \sum_{t=0}^{\infty} \beta^t \mathbb{E}_0 (u(c_t, l_t)) \\
& + \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \gamma_t \left[\mathbb{E}_t \sum_{j=0}^{\infty} \beta^t u_c(g^{t+j}) s_{t+j} - u_c(g^t) b_t \right] \\
& + \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \nu_t^L \left[\mathbb{E}_t \sum_{j=0}^{\infty} \beta^t u_c(g^{t+j}) s_{t+j} - u_c(g^t) \underline{B} \right] \\
& - \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \nu_t^H \left[\mathbb{E}_t \sum_{j=0}^{\infty} \beta^t u_c(g^{t+j}) s_{t+j} - u_c(g^t) \bar{B} \right]
\end{aligned}
$$

- $\gamma_t(g^t) < 0$ Planner would to increase indebtedness $(b_t(g^{t-1}))$ if could relax the budget constraint. Opposite when $\gamma_t(g^t) > 0$

This due to the fact that the planner cannot allocate it's debt efficiently across states.

Lagrangian 2

Using the law of iterated expectations and Abel summation formula we can express the problem with the transformed Lagrangian

$$L = \sum_{t=0}^{\infty} \beta^t \mathbb{E}_0(u(c_t, l_t))$$
$$+ \mathbb{E}_0 \beta^t \left\{ \mu_t u_c(g^t) s_t(g^t) - \gamma_t(g^t) u_c(g^t) b_t(g^{t-1}) + \nu_t^H(\bar{B} - b_t) - \nu_t^L(\underline{B} - b_t) \right\}$$

where $\mu_t = \mu_{t-1} + \gamma_t - \nu_t^H + \nu_t^L$, starting from $\mu_{-1} = 0$.

Implications 1: History dependence

Optimality for consumption:

$$u_c(g^t) - u_l(g^t) + \mu_t \left\{ (u_{cc}(g^t) - u_{cl}(g^t)) g_t(g^t) + u_c(g^t) s_{c,t}(g^t) \right\}$$

$$- \gamma_t (u_{cc}(g^t) - u_{lc}(g^t)) b_t = 0$$

Implications 1: History dependence

Optimality for consumption:

$$u_c(g^t) - u_l(g^t) + \mu_t \{(u_{cc}(g^t) - u_{cl}(g^t))g_t(g^t) + u_c(g^t)s_{c,t}(g^t)\}$$

$$- \gamma_t(u_{cc}(g^t) - u_{lc}(g^t))b_t = 0$$

- Complete markets model is equivalent to: $\mu_t = \gamma_0(\Lambda)$ and $\gamma_t = 0$ for $t > 0$ and $\nu_t = 0 \; \forall t$.

Implications 1: History dependence

Optimality for consumption:

$$u_c(g^t) - u_l(g^t) + \mu_t \{(u_{cc}(g^t) - u_{cl}(g^t))g_t(g^t) + u_c(g^t)s_{c,t}(g^t)\}$$

$$- \gamma_t(u_{cc}(g^t) - u_{lc}(g^t))b_t = 0$$

- Complete markets model is equivalent to: $\mu_t = \gamma_0(\Lambda)$ and $\gamma_t = 0$ for $t > 0$ and $\nu_t = 0 \; \forall t$.
- With incomplete markets get history dependence through μ_t and b_t. The state vector is $X = \{g_t, b_t, \mu_{t-1}\}$. Even if g_t is iid, taxes and allocations are autocorrelated.

Implications 1: History dependence

Optimality for consumption:

$$u_c(g^t) - u_l(g^t) + \mu_t \left\{ (u_{cc}(g^t) - u_{cl}(g^t)) g_t(g^t) + u_c(g^t) s_{c,t}(g^t) \right\}$$

$$- \gamma_t (u_{cc}(g^t) - u_{lc}(g^t)) b_t = 0$$

- Complete markets model is equivalent to: $\mu_t = \gamma_0(\Lambda)$ and $\gamma_t = 0$ for $t > 0$ and $\nu_t = 0 \; \forall t$.
- With incomplete markets get history dependence through μ_t and b_t. The state vector is $X = \{g_t, b_t, \mu_{t-1}\}$. Even if g_t is iid, taxes and allocations are autocorrelated.
- Term $\nu_t (u_{cc}(g^t) - u_{lc}(g^t))$ reflects planner's incentives to manipulate interest rates in order to relax the implementability constraint. More relevant with long-term bonds.

Implications 2: μ_t - risk adjusted random walk

Optimality for bonds:

$$\begin{aligned}\mu_t &= \mathbb{E}_t \frac{u_c(g^{t+1})}{u_c(g^t)} \mu_{t+1} + \nu_t^H - \nu_t^L \\ &= \mathbb{E}_t \mu_t + COV_t(u_c(g^{t+1}), \mu_{t+1})/u_c(g^t) + \nu_t^H - \nu_t^L\end{aligned}$$

μ_t follows a risk-adjusted random walk. What does this mean?

Implications 2: μ_t - risk adjusted random walk

Optimality for bonds:

$$\begin{aligned}\mu_t &= \mathbb{E}_t \frac{u_c(g^{t+1})}{u_c(g^t)} \mu_{t+1} + \nu_t^H - \nu_t^L \\ &= \mathbb{E}_t \mu_t + COV_t(u_c(g^{t+1}), \mu_{t+1})/u_c(g^t) + \nu_t^H - \nu_t^L\end{aligned}$$

μ_t follows a risk-adjusted random walk. What does this mean?
Under IM the multiplier is not constant. It measures the burden from distortionary taxation. When g_t increases μ_t increases, reflecting that taxes have to be increased.

The random walk says that its best to make this increase permanent! The reason is that tax distortions in this economy are convex... Therefore the DW loss of taxes is reduced when the tax burden is spread across periods!.

Quasilinear preferences: convergence to first-best

Let $u(c,l) = c + Blog(l)$ and $\bar{M}, \underline{\mathrm{M}}$ are equal to the natural borrowing limits, then:

$$\mu_t = \mathbb{E}_t \mu_{t+1}$$

Quasilinear preferences: convergence to first-best

Let $u(c,l) = c + Blog(l)$ and $\bar{M}, \underline{\mathrm{M}}$ are equal to the natural borrowing limits, then:

$$\mu_t = \mathbb{E}_t \mu_{t+1}$$

Claim:
If g_t is sufficiently stochastic, utility is quasilinear, under natural borrowing limits economy converges to first best! (μ_t converges to 0.)

Proof based on Doob's (1953) convergence theorem.

What does this mean?

Quasilinear preferences: convergence to first-best

Let $u(c,l) = c + Blog(l)$ and $\bar{M}, \underline{\mathrm{M}}$ are equal to the natural borrowing limits, then:

$$\mu_t = \mathbb{E}_t \mu_{t+1}$$

Claim:
If g_t is sufficiently stochastic, utility is quasilinear, under natural borrowing limits economy converges to first best! (μ_t converges to 0.)

Proof based on Doob's (1953) convergence theorem.

What does this mean?
Ramsey planner achieves complete insurance and 0 labor taxes in the long run by accumulating assets up to the natural limit.

Relaxing the assumptions

- Add-hoc borrowing limits:

  Natural limit defined $\bar{M} = \frac{g^{max}}{1-\beta}$
  When debt limits are tighter, cannot achieve $\tau = 0$, no convergence!

- Non-quasilinear utility: precautionary motive to accumulate assets. An extra reason to accumulate assets, but no proof...

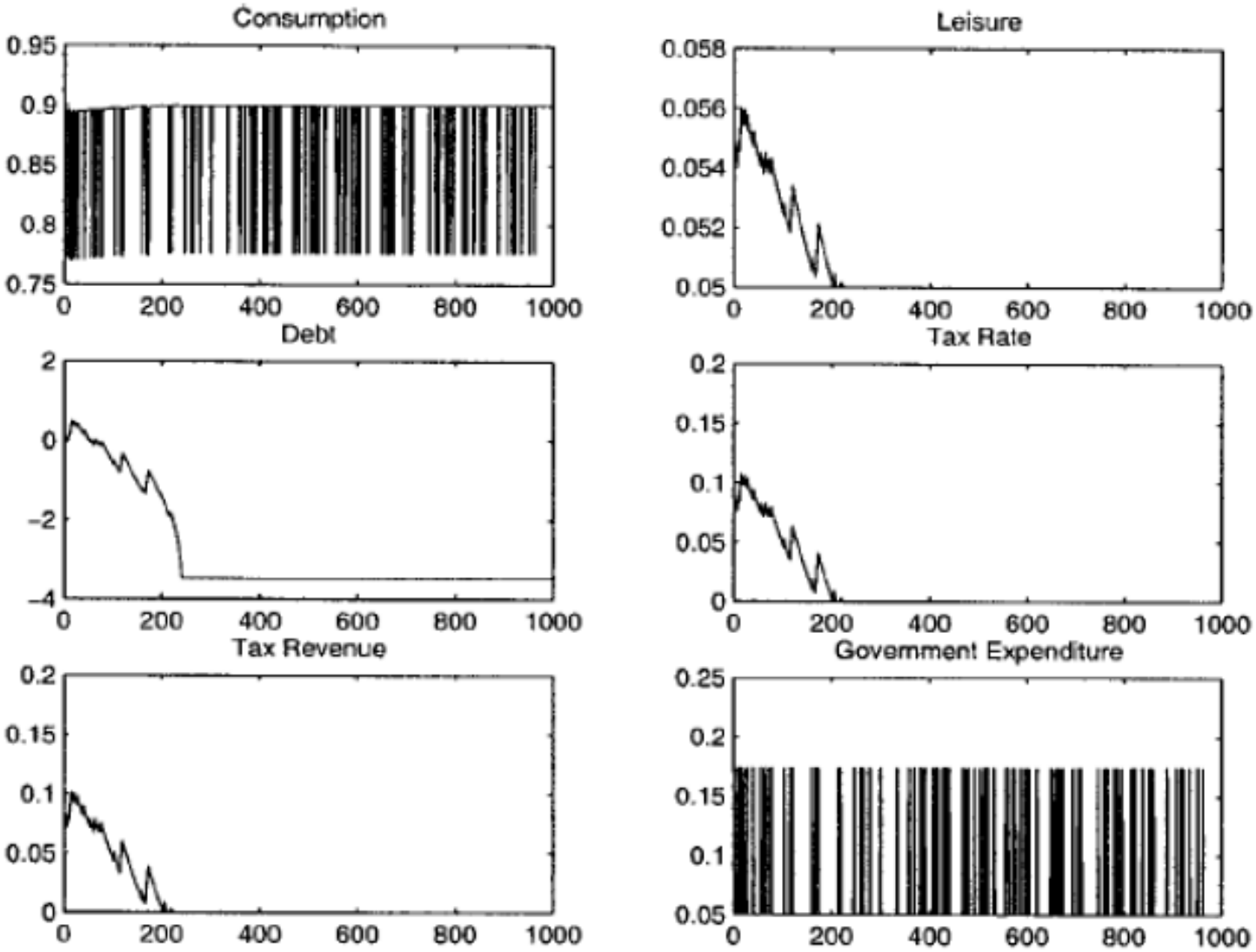
Aiyagari et. al. 2002 example

- $u(c,l) = c + 0.05log(l)$.
- $g_t \in \{0.05, 0.173\}$ (peace, war).

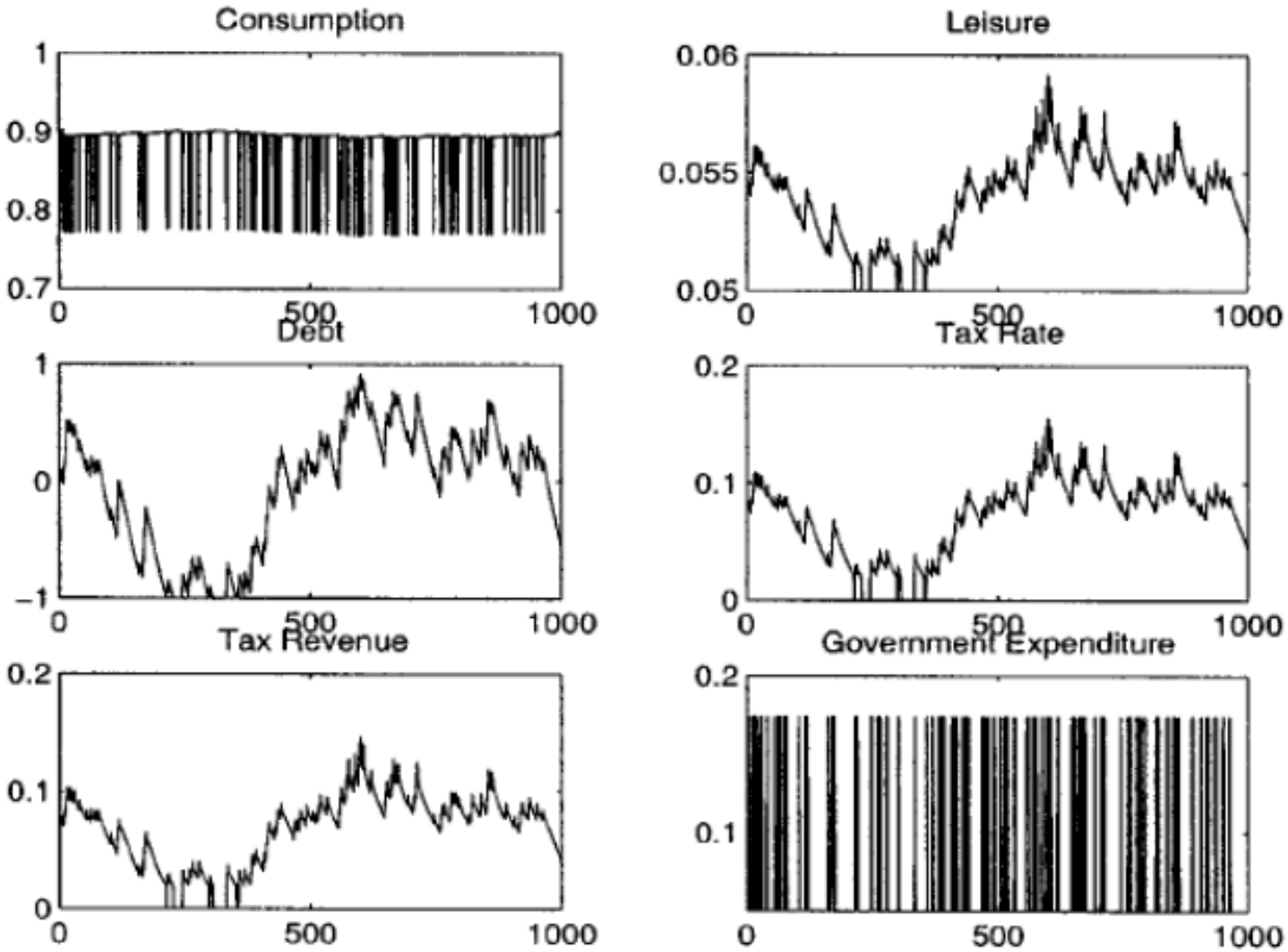
$$\Pi = \begin{pmatrix} 0.5 & 0.5 \\ 0.1 & 0.9 \end{pmatrix}$$

- Case 1. $\underline{\mathrm{M}}, \bar{M} = (-3.478, 8.58)$, natural limit.
- Case 2. $\underline{\mathrm{M}}, \bar{M} = (-1, 1)$, very tight.

Case 1: Natural borrowing limit



Case 2: Add-hoc borrowing limit



Fiscal policy with long-term bonds

- Households:

$$c_t + q_t b_{t+1}^N = (1 - \tau_t) w_t n_t + b_t^N$$

  Optimality:

$$q_t^N = \beta^N \mathbb{E}_t \frac{u_c(g^{t+N})}{u_c(g^t)} \qquad q_t^{N-1} = \beta^{N-1} \mathbb{E}_t \frac{u_c(g^{t+N-1})}{u_c(g^t)} \qquad (1 - \tau_t) w_t = \frac{u_l(g^t)}{u_c(g^t)}$$

- Government:

$$b_{t+1}^N q_t^N + \tau_t w_t n_t = g_t + q_t^{N-1} b_t$$

- Firms: $w_t = 1$

- Borrowing limits: $\underline{\mathrm{B}} \leq b_{t+1}^N \leq \bar{B}$

Fiscal policy with long-term bonds

- Governments issues large quantities of long-term debt) 10-year notes and 30-year bonds in the US)
- Used widely in research using alternative modeling assumptions (decaying coupons, constant coupons, redeemable only at the maturity etc.)

It does not matter in **Complete markets**.

But make a difference with **Incomplete markets**

Fiscal policy with long-term bonds

Bonds optimality:

$$\mu_t = \mathbb{E}_t \frac{u_c(g^{t+N})}{u_c(g^t)} \mu_{t+1} + \nu_t^H - \nu_t^L$$

Consumption optimality:

$$u_c(g^t) - u_l(g^t) + \mu_t \left\{ (u_{cc}(g^t) - u_{cl}(g^t)) s_t(g^t) + u_c(g^t) s_{c,t}(g^t) \right\}$$

$$+ (\mu_{t-N} - \mu_{t-N+1}) b_{t-N}^N (u_{cc}(g^t) - u_{lc}(g^t)) = 0$$

Term $(\mu_{t-N} - \mu_{t-N+1}) b_{t-N}^N$ reflects the interest rate manipulation. When g_t goes up, want to alleviate the implementability constraint by promising to increase consumption at $t+N$. These promises need to be respected.

Hence the relevant state space: $X = (g_t, b_t, ...b_{t-N}, \lambda_{t-1}, \lambda_{t-N})$

Example of tax manipulation

Assume complete markets. Implementability constraint:

$$b_{-1}^{N} p_0^{N-1} = \sum_{t=0}^{\infty} \beta^t \frac{s_t}{u_{c,0}}$$

Rewritten as:

$$b_{-1}^{N} u_c^{N-1} = \sum_{t=0}^{\infty} \beta^t s_t$$

if $b_{-1}^{N} > 0$, funding costs can be reduced if we increase c_{N-1}. This is done by promising to decrease taxes in $N-1$.
Therefore: $\tau_t = \bar{\tau} \quad \forall t \neq N-1 \quad$ and $\quad \tau_{N-1} < \bar{\tau}$

With complete markets this is relevant at period 0 only. When markets are incomplete, incentives for interest rate manipulation occur at every period!

How do we solve it?

We typically solve it with a version of PEA. We will use PEA (den Haan and Marcet (1990), Marcet and Lorenzoni (1999), FMOS (2014, 2019)), Valaitis and Villa (2021) to approximate the equilibrium numerically.

Grid-based methods becomes infeasible in this context. For example, if we can long-term debt with N=10, we have 21 state variable!

The goal is to approximate the expected value terms in the optimality conditions with a flexible function of the relevant state variables.

Algorithm

Algorithm

1. Generate a sequence of shocks for g_t up to $t = T$.
2. Parameterize the expected value terms in terms of the state variables X

Algorithm

1. Generate a sequence of shocks for g_t up to $t = T$.
2. Parameterize the expected value terms in terms of the state variables X

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx \Psi(X_t, \gamma_1)$$
$$\mathbb{E}_t u'(c_{t+N}) \approx \Psi(X_t, \gamma_2)$$
$$\mathbb{E}_t u'(c_{t+N-1}) \approx \Psi(X_t, \gamma_3)$$

Algorithm

1. Generate a sequence of shocks for g_t up to $t = T$.
2. Parameterize the expected value terms in terms of the state variables X

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx \Psi(X_t, \gamma_1)$$
$$\mathbb{E}_t u'(c_{t+N}) \approx \Psi(X_t, \gamma_2)$$
$$\mathbb{E}_t u'(c_{t+N-1}) \approx \Psi(X_t, \gamma_3)$$

3. Set upper and lower bond limits at some narrow interval $\underline{\mathrm{M}} < b_t^N < \bar{M}$.

Algorithm

1. Generate a sequence of shocks for g_t up to $t = T$.
2. Parameterize the expected value terms in terms of the state variables X

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx \Psi(X_t, \gamma_1)$$
$$\mathbb{E}_t u'(c_{t+N}) \approx \Psi(X_t, \gamma_2)$$
$$\mathbb{E}_t u'(c_{t+N-1}) \approx \Psi(X_t, \gamma_3)$$

3. Set upper and lower bond limits at some narrow interval $\underline{\mathrm{M}} < b_t^N < \bar{M}$.
4. For every $1 \leq t \leq T$, solve the system of optimality conditions to find c_t,b_t^N,λ_t, respecting $\underline{\mathrm{M}}$ and $\bar{M}$.

Algorithm

1. Generate a sequence of shocks for g_t up to $t = T$.
2. Parameterize the expected value terms in terms of the state variables X

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx \Psi(X_t, \gamma_1)$$
$$\mathbb{E}_t u'(c_{t+N}) \approx \Psi(X_t, \gamma_2)$$
$$\mathbb{E}_t u'(c_{t+N-1}) \approx \Psi(X_t, \gamma_3)$$

3. Set upper and lower bond limits at some narrow interval $\underline{\mathrm{M}} < b_t^N < \bar{M}$.
4. For every $1 \leq t \leq T$, solve the system of optimality conditions to find c_t,b_t^N,λ_t, respecting $\underline{\mathrm{M}}$ and $\bar{M}$.
5. Given the equilibrium sequences for b_t^N, λ_t and c_t, calculate the expected value terms, update approximation parameters $\gamma_1, \gamma_2, \gamma_3$.

Algorithm

1. Generate a sequence of shocks for g_t up to $t = T$.
2. Parameterize the expected value terms in terms of the state variables X

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx \Psi(X_t, \gamma_1)$$
$$\mathbb{E}_t u'(c_{t+N}) \approx \Psi(X_t, \gamma_2)$$
$$\mathbb{E}_t u'(c_{t+N-1}) \approx \Psi(X_t, \gamma_3)$$

3. Set upper and lower bond limits at some narrow interval $\underline{\mathrm{M}} < b_t^N < \bar{M}$.
4. For every $1 \leq t \leq T$, solve the system of optimality conditions to find c_t,b_t^N,λ_t, respecting $\underline{\mathrm{M}}$ and $\bar{M}$.
5. Given the equilibrium sequences for b_t^N, λ_t and c_t, calculate the expected value terms, update approximation parameters $\gamma_1, \gamma_2, \gamma_3$.
6. Increase $\underline{\mathrm{M}} = max(\underline{\mathrm{B}}, \underline{\mathrm{M}}) - step$ and $\bar{M} = min(\bar{M}, \bar{B}) + step$

Algorithm

1. Generate a sequence of shocks for g_t up to $t = T$.
2. Parameterize the expected value terms in terms of the state variables X

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx \Psi(X_t, \gamma_1)$$
$$\mathbb{E}_t u'(c_{t+N}) \approx \Psi(X_t, \gamma_2)$$
$$\mathbb{E}_t u'(c_{t+N-1}) \approx \Psi(X_t, \gamma_3)$$

3. Set upper and lower bond limits at some narrow interval $\underline{\mathrm{M}} < b_t^N < \bar{M}$.
4. For every $1 \leq t \leq T$, solve the system of optimality conditions to find c_t,b_t^N,λ_t, respecting $\underline{\mathrm{M}}$ and $\bar{M}$.
5. Given the equilibrium sequences for b_t^N, λ_t and c_t, calculate the expected value terms, update approximation parameters $\gamma_1, \gamma_2, \gamma_3$.
6. Increase $\underline{\mathrm{M}} = max(\underline{\mathrm{B}}, \underline{\mathrm{M}}) - step$ and $\bar{M} = min(\bar{M}, \bar{B}) + step$
7. If $\bar{M} = \bar{B}$ and $\underline{\mathrm{M}} = \underline{\mathrm{B}}$, check if parameterized expectations $\Psi(X, \gamma)$ the updating are close enough to equilibrium values for $\mathbb{E}_t u'(c_{t+N})\lambda_{t+1}, \mathbb{E}_t u'(c_{t+N}), \mathbb{E}_t u'(c_{t+N-1})$ and check if the coefficients γ are stable. If not, go back to step 4.

Optimality conditions

Consumption optimality:

$$u_c(g^t) - u_l(g^t) + \mu_t \left\{ (u_{cc}(g^t) - u_{cl}(g^t)) s_t(g^t) + u_c(g^t) s_{c,t}(g^t) \right\} + (\mu_{t-N} - \mu_{t-N+1}) b_{t-N}^N (u_{cc}(g^t) - u_{lc}(g^t)) = 0$$

Bonds optimality:

$$\mu_t = \mathbb{E}_t \frac{u_c(g^{t+N})}{u_c(g^t)} \mu_{t+N} + \nu_t^H - \nu_t^L$$

Government budget constraint:

$$g_t + b_t^N \beta^{N-1} \frac{\mathbb{E}_t u_c(g^{t+N-1})}{u_c(g^t)} = \left(1 - \frac{u_l(g^t)}{u_c(g^t)}\right) n_t + b_t^N \beta^N \frac{\mathbb{E}_t u_c(g^{t+N})}{u_c(g^t)}$$

How to parameterize?

Use a polynomial or a neural network:

Polynomial:

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx \gamma_0 + \sum_{j=1}^{N} \gamma_j \frac{b_{t-j}^N}{\bar{M}} + \gamma_{N+1}\frac{g-\bar{g}}{\bar{g}} + \sum_{j=1}^{N} \gamma_{j+N+1}\mu_{t-j}$$

Pros: relatively few parameters.
Cons: Need to commit to a functional form, variables tern to be highly correlated, so need to apply additional steps (see Faraglia et. al. 2017).

Neural Network:

$$\mathbb{E}_t u'(c_{t+N})\lambda_{t+1} \approx (ANN(X_t))$$

Pros: Robust to multicollinearity problem, don't need to commit to a functional form.
Cons: Many parameters so may a lot of data (large T).

Other details

- How to initialize γ_i? The bigger the model, the more this matters!
  Non-exhaustive list:
  - Deterministic model
  - Simulate assuming $\mathbb{E}()$ is equal to some steady state.
  - Assume $b_t = 0$ and find other variables from the optimality conditions.

- Role of $\underline{\mathrm{M}}, \bar{M}$? Most of the time, initial guess of γ_i is quite far from the true value.
  - Our guess for γ_i could imply unstable dynamics.
  - In theory b_t^N does not have the steady state and we ex-ante do not know where it needs to be. Setting tight initial $\underline{\mathrm{M}}, \bar{M}$ allows to discover this gradually. See Maliar & Maliar (2003) for reference.