Solving Khan and Thomas (2008) Using Winberry's Algorithm

Prepared by Ding Dong

Sept 2020

1 Model: Khan and Thomas (2008)

The model of Khan and Thomas (2008) belongs to a large class of macro models with heterogeneous firms. In the model, the distribution of firms over idiosyncratic states $\{\varepsilon, k\}$ has non-trivial role in shaping aggregate economy.

Household

There is a representative household whose preferences are represented by the utility function

$$\max E_t \sum_{t=0}^{\infty} \beta^t [\frac{C_t^{1-\sigma}-1}{1-\sigma} - \chi \frac{N_t^{1+\alpha}}{1+\alpha}]$$

The household owns all the firms in the economy and markets are complete.

Firms

The firm produces with a decreasing return to scale technology using capital and labor as input,

$$y_{jt} = e^{z_t} e^{\varepsilon_{jt}} k_{jt}^{\theta} n_{jt}^{\nu} \tag{1}$$

where z_t is aggregate productivity shock and ε_{jt} is idiosyncratic shock, both of which follow AR(1). $\theta + \nu < 1$.

The adjustment of capital may incur a fixed cost. The law of motion is $k_{jt+1} = (1-\delta)k_{jt} + i_{jt}$, where i_{jt} is new investment. If $\frac{i_{jt}}{k_{jt}} < -a$ or $\frac{i_{jt}}{k_{jt}} > a$, the firm must pay additional ξ_{jt} unit of labor, which is a stochastic and i.i.d from a uniform distribution over $[0, \bar{\xi}]$.

Incorporate household's problem into firm's optimization, the Bellman equation of firms is thus characterized by

$$v(\varepsilon, k, \xi; s) = \lambda(s) max_n \{y_{jt} - w(s)n\} + \max\{v^a(\varepsilon, k, \xi) - \xi\lambda(s)w(s), v^n(\varepsilon, k, \xi)\} \tag{2}$$

where marginal utility of consumption $\lambda(s) = C(s)^{-\sigma}$, $v^a(\varepsilon, k, \xi)$ denote value if a firm pays adjustment cost and invest:

$$v^a(\varepsilon, k, \xi) = \max_{k' \in R} \lambda(s)[(1-\delta)k - k'] + \beta E[v(\varepsilon', k'; \hat{s}'|\varepsilon, k; s)] \tag{3}$$

where $v(\varepsilon,\hat{k};s) = intv(\varepsilon',k';s'|\varepsilon,k;s)dG(\xi)$ and $v^n(\varepsilon,k,\xi)$ denote value if a firm invest within the rate [-a,a]:

$$v^n(\varepsilon,k,\xi) = \max_{k'\in[(1-\delta-a)k,(1-\delta+a)k]} \lambda(s)[(1-\delta)k-k'] + \beta E[\int v(\varepsilon',k';s'|\varepsilon,k;s)dG(\xi)] \quad (4)$$

There will be a unique threshold value of fixed cost $\xi(\varepsilon,k)$ making the firm indifferent between unconstrained or constrained investment:

$$\xi(\varepsilon,\tilde{k};s) = \frac{v^a - v^n}{\lambda(s)w(s)} \quad (5)$$

Therefore, the expectation over ξ can be expressed analytically:

$$v(\varepsilon,\hat{k};s) = \lambda(s)\max_n\{y_{jt} - w(s)n\} + \frac{\hat{\xi}}{\bar{\xi}}[v^a(\varepsilon,k;s) - \lambda(s)w(s)\frac{\xi(\varepsilon,\hat{k},\xi)}{2}] + (1-\frac{\hat{\xi}}{\bar{\xi}})v^n(\varepsilon,k;s) \quad (6)$$

Equilibrium

- firm: solve problem above;
- household: $\lambda(s) = C(s)^{-\sigma}$
- output market clearing

$$C(s) = \int [y + (1-\delta)k - \frac{\hat{\xi}}{\bar{\xi}}k^a(\varepsilon,k;s) + (1-\frac{\hat{\xi}}{\bar{\xi}})k^n(\varepsilon,k;s)]$$

- labor market clearing

$$\int [n(\varepsilon,k;s) + \frac{\hat{\xi}^2(\varepsilon,k;s)}{2\bar{\xi}}]g(\varepsilon,k)d\varepsilon dk = [\frac{w(s)\lambda(s)}{\chi}]^{\frac{1}{\alpha}}$$

- law of motion for distribution

$$g'(\varepsilon',k';s') = \int\int [\frac{\hat{\xi}}{\bar{\xi}}\{k^a(\varepsilon,k;s) = k'\} + (1-\frac{\hat{\xi}}{\bar{\xi}})\{k^n(\varepsilon,k;s) = k'\}]f(\varepsilon'|\varepsilon)d\varepsilon dk \quad (7)$$

- law of motion for aggregate productivity

$$z' = \rho_z z + \sigma_z \omega_z' \quad (8)$$

Introduction to the Algorithm

The algorithm of Winberry (2018) involves three major steps:

- Approximate equilibrium objects using finite-dimensional global approximation with respect to individual state variables;
- Compute approximated stationary equilibrium with idiosyncratic shocks, but no aggregate shock;
- Introduce aggregate shock and do perturbation around approximated stationary equilibrium.

2 Solving the Steady State

This script will solve for the steady state of the model, plot decision rules, plot the stationary distribution, compare the parametric family approximation of the distribution to a non-parametric histogram, and compute steady state aggregates. See Appendix A.2 of the paper for a discussion of how to solve the steady state.

set up parameters (setParameters.m)

- set up model parameters;
- set up approximation parameters:
  - approximation of value function (order of n and n_k)
  - grid of idiosyncratic states (and k)
  - parameters of value function iteration (max. steps, tolerance, Howard improvement steps, damping)
  - approximation of distribution (order of n_g: $nMeasure$, # of quadrature points on each dimension, # of coefficient g: $nMeasureCoefficients$)

Compute grids in various approximation (computeGrids.m)

including

- nodes to integrate idiosyncratic shocks (i.e. $vShocksGrid$) and weights (i.e. $vShocksWeights$) (computeGaussHermiteQuadrature.m)
- grids for approximating value function and capital accumulation decision conditional on adjusting (Chebyshev collocation nodes)
  - compute original Chebyshev polynomials for (ε, k) within [-1,1];
  - compute tensor product grid (i.e. $mStateGridZeros$)
  - scale up to state space (scaleUp.m)
  - compute grid of future productivity shocks (for computing expectations) (scaleDown.m)
- grids for approximating histogram and plotting functions
  - standard grid (finer than Chebyshev collocation nodes)
  - individual state grids
  - compute tensor product grid
  - scale down to [-1,1] for polynomials (scaleDown.m)
  - compute grid of future productivity shocks
  - compute Tauchen transition matrix for productivity shocks
- nodes and weights to integrate parametric family
  - compute grids in the interval [-1,1] (computeGaussHermiteQuadrature.m)
  - scale up grid (scaleUp.m)
  - compute tensor product grid
  - scale down to [-1,1] (scaleDown.m)
  - compute tensor product weights
  - compute grid over future productivity shocks (useful in computing decisions)

Compute polynomial for approximating various objects (computePolynomials.m)

including

- polynomials for value function and capital accumulation conditional on adjusting
  - create one-dimensional polynomials from computeGrids.m (i.e. $mStateGridZeros$)
  - compute tensor product in polynomials
  - compute squared terms for interpolation formulas (computeChebyshev.m)
  - compute polynomials over future productivity shocks
- Chebyshev polynomials over fine grid for approximating histogram and plotting functions
  - create one-dimensional polynomials (computeChebyshev.m)
  - compute tensor product of polynomials
  - compute polynomials over future shocks
- Chebyshev polynomials over quadrature grid for computing law of motion for parametric family
  - create one-dimensional polynomials (computeChebyshev.m)
  - compute tensor product of polynomials
  - compute polynomials over future shocks
- Derivative of Chebyshev polynomials over collocation nodes for computing first-order condition in dynamic model
  - create one-dimensional polynomials (computeChebyshev.m)
  - compute tensor product of polynomials
- Derivative of Chebyshev polynomials over fine grid for plotting marginal value function
  - create one-dimensional polynomials (computeChebyshev.m)
  - compute tensor product of polynomials

Solve initial guess for w^* using histogram (computeLMCResidualHistogram.m)

- use an initial guess on wage (i.e. $wRepSS$) as input
- compute implied labor demand residual
  - compute labor demand from w=MPL on the grid (ε, k)
  - compute profit, i.e. output less wage bill, on the grid (ε, k)
  - compute parameters of approximation of value function θ_{ij} (i.e. $vCoefficients$) using value function iteration (updateCoefficients.m)
    - compute investment decision conditional on adjustment (i.e. $vCapitalAdjust$) [1] (capitalResid.m)
    - compute investment decision conditional on no adjustment (i.e. $vCapitalConstrained$)
    - compute polynomials of next period's value function, conditional on investment decisions (i.e. $mCapitalAdjustPrimePoly$ and $mCapitalConstrainedPrimePoly$)
    - compute expected value function, if adjust (i.e. $vValueAdjust$) or not (i.e. $vValueConstrained$)
    - compute cut-off adjustment cost $\hat{\xi}$ (i.e. $vCutoff$) and RHS of Bellman equation (i.e. $vNewGrid$)
    - compute new coefficients (i.e. $vCoefficients$) and update for finite times
  - compute polynomial approximation of capital accumulation policy conditional on adjustment (updateCoefficients.m)
  - compute polynomial approximation of capital accumulation policy conditional on adjustment or not (computePolicies.m)
  - compute capital accumulation policy functions and cut-off shock over fine grid (computePolicies.m)
  - compute stationary distribution from policy functions using histogram following Young (2010) (computeDiscreteTransitionMatrix.m)
  - compute aggregate demand at labor market and residual
- solve for market clearing wage (i.e. $wageInit$)
- compute value function, policy function and distribution (i.e. $vHistogram$) at $w = wageInit$ (computeLMCResidualHistogram.m)

[1] only do it once per productivity value since its independent of capital

Compute moments of $vHistogram$ as initial guess for parametric family (coreSteadyState.m)

- compute 1st, 2nd and higher moments from $vHistogram$ (i.e. $vMomentsHistogram$)
- compute grid of centralized moments from moments (i.e. $mGridMoments$)
- compute parameters (i.e. $vParameters$) by minimizing integral equation[2] (parametersResidual.m)

Solve refined wage using exponential polynomials[3] (coreSteadyState.m)

- use an initial guess on wage (i.e. $wageInit$) and initial guess of distribution parameters (i.e. $vParmaters$, $vMoments$, $mGridMoments$) as input
- compute implied labor demand residual (computeLMCResidualPolynomials.m)
  - compute labor demand from w=MPL on the grid (ε, k)
  - compute profit, output less wage bill, on the grid (ε, k)
  - compute parameters of approximation of value function (θ_{ij}) using value function iteration (updateCoefficients.m)
  - compute polynomial approximation of capital accumulation policy conditional on adjustment (updateCoefficients.m)
  - compute polynomial approximation of capital accumulation policy conditional on no adjustment (computePolicies.m)
  - compute capital accumulation policy functions and cut-off shock over fine grid (computePolicies.m)
  - enforce choices to be within bounds
  - replicate choices for each draw of future idioysncratic shock
  - compute stationary distribution from policy functions by iterating on law of motion
  - compute aggregate demand at labor market and residual
  - compute compute steady state aggregates (i.e. aggregate consumption, marginal utility, output, capital, investment)
- solve for market clearing wage (i.e. $wage$)

[2] g_0 is chosen so that the total mass of the p.d.f. is 1.

[3] I use blue-colored text to highlight difference from previous step using *histogram*.

Compute steady state objects over histogram grid for plots

- compute compute steady state aggregates (i.e. aggregate consumption, marginal utility, output, capital) (computeLMCResidualHistogram.m)
- compute capital accumulation policy conditional adjustment or not (computePolicies.m)
- compute marginal value function of capital (i.e. $vCoefficientsDeriv$)

Compute aggregates variables (computeLMCResidualPolynomials.m)

including

- parameters of distribution (i.e. $vParameters$)
- moments (i.e. $vMoments$)
- aggregate consumption (i.e. $aggregateConsumption$)
- marginal utility (i.e. $marginalUtility$)
- output (i.e. $aggregateOutput$)
- capital (i.e. $aggregateCapital$)
- investment (i.e. $aggregateInvestment$)

Compute density from parametric family over fine grid

- compute fine grid of centered moments (i.e. $mGridMomentsFine$) with moments (i.e. $vMoments$) solved above;
- compute distribution over fine grid (i.e. $mFineDistribution$) with parameters (i.e. $vParameters$) solved above;

Plot and analyze steady state objects

3 Computing the Dynamic Model

This code solves for a local approximation of the model's dynamics using DYNARE. DYNARE will automatically print time-series statistics and plot impulse responses.

Set up (similar to steady state part)

- set parameters (setParameters.m)
- set grids (computeGrids.m)
- set polynomials over grids (computePolynomials.m)
- save above as parameters for DYNARE (i.e. *economicParameters.mat*, *approximationParameters.mat*, *grids.mat*, *polynomials.mat*)

Solve for aggregate dynamics using DYNARE (dynamicModel.mod)

using Macro processors to

- load parameters[4] (@include "parameters.mod")
  - load in .mat files containing parameters (i.e. *economicParameters.mat*, *approximationParameters.m*
    grids.mat, *polynomials.mat*)
  - define economic parameters (as in standard Dynare .mod file)
  - assign values of economic parameters (from *economicParameters.mat*)
  - define approximation parameters
  - assign values of approximation parameters (from *approximationParameters.mat*)
  - define value function grid
  - assign values of value function grid (from *grids.mat*)
  - define quadrature grid and weights for idiosyncratic shock, integrating measure, future productivity
  - assign value of quadrature nodes and weights (from *grids.mat*)
  - define polynomial over state grid, over future productivity, derivative of value function, squared terms of Chebyshev interpolation, over quadrature grid
  - assign values of polynomial (from *polynomials.mat*)

[4] Parameters include the grids and polynomials defined over the grids.

- define variables (@include "variables.mod")
  - value function coefficients (θs)
  - capital policy conditional on adjustment or not
  - moment
  - distribution parameters (gs)
  - prices: wage and marginal utility
  - aggregate shock (TFP)
  - other variables of interest, including aggregateConsumption, aggregateHours, expectedMarginalUtilityPrime, realInterestRate, logAggregateOutput, logAggregateConsumption, logAggregateInvestment, logAggregateHours, logWage, logMU
  - exogenous shocks
- model equations (@include "equations.mod")
  - expand capital policy conditional on adjustment (i.e. *capitalAdjust*) along entire grid (# of equations = nState= nProd*nCapital)
  - define Bellman equation for each point in iState in the individual state space
    - compute expected value for next period, conditional on capital adjustment or not (and the cut-off)
    - compute RHS of Bellman equation: flow profits + expected value function
  - FOC for adjust capital decision with MP*MU= MC (# of equations = nProd)
  - compute objects over quadrature grid for integrating distribution
  - relationship between moments of distribution and parameters (# of equations = nMeasureCoefficients)
    - define first moments (uncentered)
    - define higher moments (centered)
  - law of motion for distribution (# of equations = nMeasureCoefficients)
    - compute first moment of productivity (uncentered)
    - compute first moment of capital (uncentered)
    - compute higher order moments (centered)
  - labor Market clearing (# of equations = 2)
    - define aggregate hours from labor demand

        * set labor demand = labor supply
    - output Market clearing (# of equations = 2)
        * define aggregate consumption
        * set marginal utility = u'(aggregate consumption)
    - law of motion for aggregate shocks (# of equations = 1)
        * AR(1) process
    - auxiliary variables of interest (# equations = n)
        * define aggregateConsumption, aggregateHours etc.
- specify the shock process
- steady state (dynamicModel_steadystate.m)
    - (computes stationary equilibrium of the model in a format required by Dynare)
    - read in parameters from Dynare declaration
    - call parameters (setParameters_steadystate.m)
    - solve for steady state wage (coreSteadyState.m)
    - save grid and polynomials
    - save steady state results for Dynare (updateCoefficients.m)
    - load output for Dynare
- simulate the economy

Reference

Khan, A. & Thomas, J. K. (2008). Idiosyncratic shocks and the role of nonconvexities in plant and aggregate investment dynamics. *Econometrica*, *76*(2), 395–436.

Winberry, T. (2018). A method for solving and estimating heterogeneous agent macro models. *Quantitative Economics*, *9*(3), 1123–1151.

Young, E. R. (2010). Solving the incomplete markets model with aggregate uncertainty using the krusell–smith algorithm and non-stochastic simulations. *Journal of Economic Dynamics and Control*, *34*(1), 36–41.