An Overlapping-Generations Model with Perfect Foresight Solved by the Sequential Recalibration Algorithm *

Kerk L. Phillips[1]

[1]US Congressional Budget Office, Washington, DC, USA

November 9, 2022
version 2022.11.d

Abstract

This paper builds a dynamic general equilibrium (DGE) overlapping-generations (OLG) model of the U.S. economy. The model is solved and simulated using the Sequential Recalibration Algorithm (SRA) proposed by Rausch and Rutherford (2009). We show that for a very simple OLG model the SRA method solves substantially faster than other available methods and does so with high accuracy. We conclude that it is an excellent candidate for solving and simulating larger scale OLG models.

keywords: dynamic stochastic general equilibrium, United States, overlapping generations, forecasting models, computational methods

JEL classifications: C63, E62, H5

IN PROGRESS

*The views expressed in this paper are the authors' and should not be interpreted as those of the Congressional Budget Office or the Federal Reserve. The Python code used in this paper is available at https://github.com/kerkphillips/OLG_SRA_code.git

1 Introduction

This paper evaluates the pros and cons of using the Sequential Recalibration Algorithm (SRA) proposed by Rausch and Rutherford (2009) to solve and simulate policy changes in an overlapping-generations (OLG) model. We compare this method to two other solution methods: the traditional time-path iteration method introduced in Auerbach and Kotlikoff (1987) (denoted as the AK method) and a more recent short-cut to that method proposed in Evans and Phillips (2014) (the AMF method). We set up a model with $S = 80$ generations, each period corresponding to a year. We also allow for J different labor productivity types in each cohort. We solve and simulate the model using all three methods and compare the accuracy and compute time for each method. We do this for several different values of J. This is because one potential advantage of the SRA is that it allows for more efficient calculation than the other two methods as the heterogeneity of model agents rises.

1.1 An Illustration of the Potential Gains

Here we briefly present and discuss an OLG model. This model is NOT the one we will use to test the relative merits of the solution methods. That model is developed in Section 2. The model here is not fleshed out in great detail, but it captures the key features used in the models from Rausch and Rutherford (2009) and Evans and Phillips (2014). Both of these studies solve similar models and compare their results with those from using the AK approach.

We are leaving the production side of the economy deliberately vague. We could have a single good. Or we could have multiple goods being produced and aggregated via the households into the aggregate consumption good c. We could also have many intermediate goods which are then aggregated into a single final good which can be consumed or invested.

There are S generations of households alive in any given period. The youngest are age 1 and the oldest are age S. Each household has an endowment of labor time that can only be supplied to the labor market in exchange for wages. The endowment may vary by age and is denoted ℓ_s. Households face idiosyncratic risk associated with their labor productivity. We assume there are J unique values for labor productivity and denote this level as a_j. Finally, each household begins a period with a stock of capital carried over from the previous period. New households of age 1 have a starting capital stock of zero. For ease of computation, it is common practice to discretize the holdings of capital. We will allow for N different values and index these with n. The capital stock held in the current period, t, by a household of age s with labor productivity level j and capital holdings corresponding to level n is denoted k_{sjnt}.

We assume that a household's labor productivity type follows a discrete Markov process with the probability of transitioning from type j today to type i tomorrow denoted as π_{ji}. The number of households holding k_{sjnt} is denoted p_{sjnt} and evolves over time according to demographic assumptions, transitioning between Markov states, and optimal savings decisions of the households.

Assuming a CRRA utility function, $u(c) = \frac{1}{1-\gamma}(c^{1-\gamma} - 1)$, a typical intertemporal Euler equation and budget constraint for a household in this economy can be written as follows.

$$c_{sjnt}^{-\gamma} = \beta \sum_{i=1}^{J} \pi_{ji} c_{s+1,i,m,t+1}^{-\gamma} (1 + r_{t+1} - \delta) \tag{1.1}$$

$$w_r a_j \ell_s + (1 + r_t - \delta) k_{sjnt} = c_{sjnt} + k_{s+1,j,m,t+1} \tag{1.2}$$

whee m indicates the optimal level of capital to carry over to next period.

All three of the methods require solving backward over time from the steady state in the final period, T We need to know what the choice of

capital holdings in the current period must be to generate the known capital holdings in the next period. We can do this a few ways.

For example, we could rewrite equations (1.1) and (1.2) to get a closed form solution. We know that the oldest households must have policy functions that set capital after death to zero. This means that consumption is given by the budget constraint.

$$c_{Sjnt} = w_t a_j \ell_S + (1 + r_t - \delta) k_{sjnt} \tag{1.3}$$

$$k_{S+1,j,n,t+1} = 0 \tag{1.4}$$

Note that this requires knowledge of an information set, Θ_t, that includes $w_t, a_j, \ell_s, r_t, \& \ \delta$.

For other ages we have:

$$c_{sjnt} =^{-\gamma}= \left[\beta \sum_{i=1}^{J} \pi_{ji} c_{s+1,i,m,t+1}^{-\gamma} (1 + r_{t+1} - \delta)\right]^{-\frac{1}{\gamma}} \tag{1.5}$$

$$k_{s+1,j,m,t+1} = c_{sjnt} - w_r a_j \ell_s - (1 + r_t - \delta) h_{sjnt} \tag{1.6}$$

Another way to find policy functions would be to use the discretized grid for k and do numerical iteration to get a discrete approximation of the equations above.

Either way, once these policy functions are known, we can easily run through the time-path simulation and determine the behavior of every household alive in every period of the simulation. We begin in the first period and solve for the lifecycle behavior for every household from now until age S by chaining the functions above appropriately. We also do the same for every household born in every period up to $T - 1$.

This requires holding the elements of $\{\Theta_t\}_{t=1}^{T}$ fixed even though these are general equilibrium values. It is the approach to updating these values that distinguished our three methods.

We must begin any solution for the time path from an initial state to the steady state by making a preliminary guess for the elements of $\{\Theta_t\}_{t=1}^{T}$, choosing a value of T large enough that we can expect the economy to asymptotically converge to the steady state by that period. Typically the elements of Θ are functions of the factors in the model. In our case, they are all functions of the aggregate capital stock series, $\{K_t\}_{t=1}^{T}$.

In the first period there are $S \times J \times N$ households alive of various ages. A household of age s will require $S - s + 1$ chained function calls to compute their lifecycle behavior. This gives $\frac{1}{2}S(S-1) \times J \times N$ function calls for the first time period. For the remaining $T-2$ time periods we need only evaluate new households, each requiring S calls. Since some of these households will not die until after T we need to extend the K series an addition S periods adding the steady state value of K for these later periods. This gives an additional $S \times (T-2) \times J \times N$ function calls. The total number of calls is $X = S[\frac{1}{2}(S-1) - (T-2)] \times J \times N$. If, for example, $S = 60$, $J = 8$, $N = 200$, and $T = 100$, the the total number of policy function calls would be $X = 12,240,000$.

The AK method requires an initial guess for the time path of K. Once we have evaluated all the policy functions we can aggregate the capital holdings of all households each period to get an new updated time path for K. We used a convex combination of this new time path and the original time path as a new guess and repeat the process. When the new and original series are sufficiently close to each other we are finished, having performed I_{AK} iterations and a total of $X \times I_{AK}$ policy function calls. Appendix A presents the algorithm in more detail.

For the SRA method we use an RA equivalent of the OLG model to forecast the time series for K. We then proceed as above to evaluate the policy choices of all households for all time periods. In this case, however, when done we examine the overall shares of goods consumptions from the OLG simulation and we recalibrate the parameters in the RA model to match

these observed shares. We then iterate again until the changes between iterations are sufficiently small. This algorithm is presented in Appendix C. Again we have $X \times I_{SRA}$ policy function calls. But it usually the case that $I_{SRA} < I_{AK}$. SRA also has some addition overhead from solving the time path each iteration for the RA model, but this addition calculation time is likely to be relatively small.

Finally, for the AMF method we again choose a simple forecasting model. This can be something as simple as linear trend between the starting value an the steady state value of K.[1] We again solve forward over time, using backward induction just as with the other two methods. However, once we have evaluated a period we save the lifecycle values only for that period. We aggregate up next periods holdings of capital to get a new starting value for a new forecast from the next period to T. This means we must treat each period as if it is the first and run $\frac{1}{2}S(S-1)\times J\times N$ function calls, and we must do this for $T-1$ periods. This gives a total of $Y = \frac{1}{2}S(S-1)\times J\times N\times(T-1)$ function calls for this method. However, despite having a larger number of calls to run through the time path, we need only do one pass. No additional iteration is required. We present this algorithm in Appendix B. For the same dimensions as above this gives $Y = 280,368,000$ calls or a little less than 23 times the number needed for a single iteration with either of the previous methods.

So, if either of the above methods takes 23 iterations or longer, then AMF would be a faster method to calculate the time path.

Evans and Phillips (2014) use values of $S = 60, J = 7, N = 350$ and $T = 70$ which implies a single pass through the AMF method should take about 20 times longer than a single pass through AK. They report compute time for AMF that is 85 percent lower than that for AK. They do not report

[1] Kotlikoff (1998), points out that related methods were used in Summers (1981) and Seidman (1983) (based on Miller and Upton (1974)). Notably, these papers assumed fixed steady state expectations over the entire transition path and were used because there was no known method for solving for the rational expectations transition path at the time.

the total number of iterations for AK, but the figures calculated above imply that the number was about 130. Their code was written in MATLAB. In addition, their model has only a single final good.

Rausch and Rutherford (2009) run a series of models with varying degrees of heterogeneity and with multiple final goods. Their code is written in GAMS, which has very efficient solvers relative to MATLAB, so comparisons of compute time are problematic. They also do not report the number of iterations for the AK solutions to their various models, but they do report iterations for their SRA runs. These numbers range from 29 to 74.

Both the SRA and AMF methods would appear to yield substantial gains in compute time relative to the traditional AK method. If the SRA method can converge in a sufficiently smaller number of iterations, it will have shorter compute times than AMF. The number of iterations necessary depends on the length of the simulation and the number of periods in a lifetime, but appears to lie somewhere in the range of 20 to 80 iterations.

We will generate more concrete comparisons for a much simpler model Section 5. First, we develop this much simpler mode in Section 2. Section 3 develops the representative agent equivalent which will be used with the SRA method. Section 4 discusses calibration of the models. And, finally the results of our model comparison 'horse race' are presented in Section 5.

2 A Very Simple OLG Model

This paper builds an OLG model with perfect foresight. The model has J ability types for worker productivity. Agents are economically active for 80 periods, and the youngest cohort roughly corresponds to a worker between the ages of 16 to 20 who is just entering the workforce.

2.1 Households

Households vary by age ($1 \leq s \leq S = 80$) and ability level ($1 \leq j \leq J$). A household of age s and ability type j solves the following dynamic program. This is a ridiculously simple model with no mortality risk, no ability variation by age, no growth of population or technology, and a simple flat income tax. There are an equal number of households of each (s, j) type, normalized to one.

$$\begin{aligned} V_{sj}(k_{sjt}; \Theta_t) = & \max_{k_{s+1,j,t+1}, \ell_{sjt}} \frac{1}{1-\gamma}(c_{sjt}^{1-\gamma} - 1) - \chi \frac{1}{1+\theta}\ell_{sjt}^{1+\theta} \\ & + \beta V_{s+1,j}(k_{s+1,j,t+1}\Theta_{t+1}) \\ c_{sjt} = & (1-\tau)\left[w_t a_{sj}\ell_{sjt} + (r_t - \delta)k_{st}\right] + k_{sjt} + f_{sjt} - k_{s+1,j,t+1} \end{aligned} \tag{2.1}$$

The Euler equations for this problem are:

$$c_{sjt}^{-\gamma} = \beta c_{s+1,j,t+1}^{-\gamma}[1 + (r_{t+1} - \delta)(1-\tau)] \tag{2.2}$$

$$c_{sjt}^{-\gamma}(1-\tau)w_t a_{sj} = \chi \ell_{sjt}^{\theta} \tag{2.3}$$

2.2 Firms

There is a single final good produced with a Cobb-Douglas production function and sold in a perfectly competitive market.

$$Y_t = K_t^{\alpha} L_t^{1-\alpha} \tag{2.4}$$

$$r_t = \alpha \frac{Y_t}{K_t} \tag{2.5}$$

$$w_t = (1-\alpha)\frac{Y_t}{L_t} \tag{2.6}$$

2.3 Government

The government balances it's budget each period by refunding all tax revenues lump-sum to the households. The lump sum transfers are allocated equally over all household types.

$$F_t = \tau \left[w_t L_t + (r_t - \delta) K_t \right] \tag{2.7}$$

$$f_{sjt} = \frac{F_t}{SJ} \tag{2.8}$$

2.4 Market Clearing

The labor and capital market clearing conditions are:

$$L_t = \sum_{s=1}^{S} \sum_{j=1}^{J} a_{sj} \ell_{sjt} \tag{2.9}$$

$$K_t = \sum_{s=1}^{S} \sum_{j=1}^{J} k_{sjt} \tag{2.10}$$

We ignore the goods market clearing condition as redundant by Walras' Law.

2.5 Additional Definitions

We define aggregate consumption and investment as follows:

$$C_t = \sum_{s=1}^{S} \sum_{j=1}^{J} c_{sjt} \tag{2.11}$$

$$I_t = K_{t+1} - (1 - \delta) K_t \tag{2.12}$$

2.6 Summary

Equations (2.1) - (2.12) define a dynamic system of $3SJ + (S-1)J + 8$ equations defining the same number of unknowns each period;

$\left\{\{c_{sjt}, \ell_{sjt}, f_{sjt}\}_{s=1}^{S}, \{k_{sjt}\}_{s=2}^{S}\right\}_{j=1}^{J}, Y_t, r_t, w_t, F_t, L_t, K_t, C_t, \& I_t.$

3 The Representative Agent Model

The SRA requires a representative agent (RA) simplification of the OLG model. In this section we build such a model by substituting a single infinitely-lived agent for the various cohorts of OLG model. This representative agent solves the following dynamic program.

$$\begin{aligned} V(K_t; \Theta_t) &= \max_{K_t, L_t} \frac{1}{1-\gamma}(C_t^{1-\gamma} - 1) - \chi \frac{1}{1+\theta} L_t^{1+\theta} \\ &\quad + \beta V(K_{t+1} \Theta_{t+1}) \\ C_t &= (1-\tau)[w_t L_t + (r_t - \delta)K_t] + K_t + F_t - K_{t+1} \end{aligned} \tag{3.1}$$

The Euler equations for this problem are:

$$C_t^{-\gamma} = \beta C_{t+1}^{-\gamma}[1 + (r_{t+1} - \delta)(1-\tau)] \tag{3.2}$$

$$C_t^{-\gamma}(1-\tau)w_t = \chi L_t^{\theta} \tag{3.3}$$

Equations (2.4) - (2.7) and (2.12) carry through from the OLG model unchanged. These equations define a dynamic system of 8 equations in the following 8 unknowns each period: $C_t, K_t, L_t, Y_t, r_t, w_t, F_t, \& I_t$.

4 Calibration and Steady State

4.1 OLG Model

We use the parameters listed in Table 1. Most of the parameters are those commonly found in dynamic general equilibrium models of the macroeconomy. We are not overly concerned with the exact values of these parameters as a careful calibration of our model to a specific economy is not the purpose

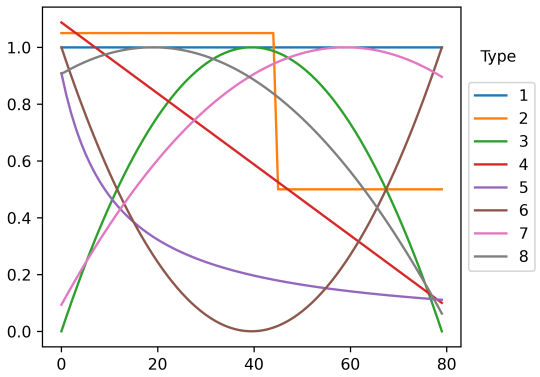
of this paper.

Table 1: List of OLG Model Parameters

S	Number of cohorts alive in a given period	80
J	Number of ability types per cohort	1 - 8
β	Subjective discount factor (annual basis)	.95
γ	CRRA and CES parameter for goods utility	3.0
θ	Inverse of Frisch elasticity	2.0
χ	Utility weight on leisure	10
α	Capital share in output	.35
δ	Depreciation rate for capital (annual basis)	.08
τ	long-run flat income tax rate	.2
a_{sj}	labor productivity by type	Figure 1

We choose the lifecycle profiles for household labor productivity (a_{sj}) to be quite different across the different ability types. Choosing productivities that simple scale the same profile shape over types does not present a substantive computational challenge as it implies that the different types differ only in the size of their effective labor and their consumption, labor, and capital profiles will be appropriately scaled versions of one another. Figure 1 plots the ability profiles for eight different types. Average labor productivity over all ages and types is .6610.

Figure 1: Labor Productivity Profiles by Age



We report the steady state values for the model with $J = 8$ in Table 2. Appendix E details the solution method for finding the steady state. In addition to the values reported in the table, we also obtain steady state lifecycle profiles for consumption, labor, and capital holdings for all eight productivity types.

Table 2: List of OLG Steady State Values

$\bar{K}$	Aggregate capital stock	941.8
$\bar{L}$	Aggregate effective labor	274.7
$\bar{Y}$	Aggregate output	422.8
$\bar{C}$	Aggregate consumption	347.5
$\bar{I}$	Aggregate investment	75.3
$\bar{F}$	Aggregate taxes / transfers	69.5
$\bar{r}$	Rental rate on capital	15.7%
$\bar{r} - \delta$	Net interest rate	7.7%
$\bar{w}$	Wage rate	1.000
$\frac{\bar{K}}{\bar{Y}}$	Capital-to-Output ratio	2.228
$\frac{\bar{C}}{\bar{Y}}$	Consumption share in GDP	.8218
$\frac{\bar{F}}{\bar{Y}}$	Taxes as percent of GDP	.1644
	Average labor supply per household	.6621

4.2 RA Model

The SRA method requires recalibration of the RA model to match key moments from the OLG model. In our case we will target: 1) the consumption-to-GDP ratio, 2) the average labor supply per household, and 3) tax revenues as a percent of GDP. We achieve these targets by appropriately altering the values of β, χ, and τ in the RA model. These targets and the associated parameters change depending on the value of J since the addition of additional productivity types changes the average labor supply. Table 3 shows the values of these three parameters needed to hit the targets for various values of J. These are only the initial calibrations of the RA model. Over the course of solving for the transition path the targets and parameters are iteratively updated.

Table 3: List of Parameter Values and Targets

	$J=1$	$J=2$	$J=4$	$J=8$	OLG
β	.940	.946	.942	.942	.95
χ	9.751	7.648	4.653	3.108	10
τ	.200	.200	.200	.200	.2
C/Y	.825	.815	.821	.822	
L	.528	.551	.611	.662	
F/Y	.165	.163	.164	.164	

5 Simulation Results

To calculate a non-trivial transition path, we must choose an initial state that is different from the steady state to which the simulation will eventually converge. For simplicity, we choose an initial state where the capital holdings of all households are one-half the steady state value. This is not a particularly realistic initial state, but the purpose of this study is to test the numerical properties of the various solution methods, not to realistically model any particular economy or policy response.

Table 4 shows the execution time, number of iterations, and accuracy for each method for four levels of heterogeneity. Execution times are relative to the AK method with a single ability type. The number of iterations is given in parentheses for AK and SRA; AMF does only one iteration. Accuracy is measured by the mean absolute percent deviation (MAPD) of the time path for K relative to that for the AK method. The levels of heterogeneity are one, two, four, and eight ability types with the age profiles of these types illustrated by Figure 1.

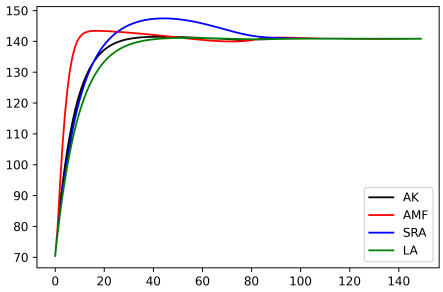
Both AMF and SRA run more quickly that AK, but SRA is several time faster than AMF. While AMF takes, on average, 81 percent of the time that AK does, SRA takes only 9.3 percent of the time. SRA is much quicker because it take so few iterations to converge. Rausch and Rutherford (2009) note that is because the most substantive change occurs in the first

iteration as the RA model parameters are adjusted to meet OLG targets, later iterations make only minor changes to these values. Both AMF and SRA have good fit with the AK solution, averaging MAPDs of 1.8 and 1.2 percent, respectively. Figure 2 shows the final paths for capital from each solution method for all four values of J. The fit with the AK solution is best as the simulation approaches the steady states, but for both AMF and SRA there are notable deviations from the AK path in the first half of the transition paths.

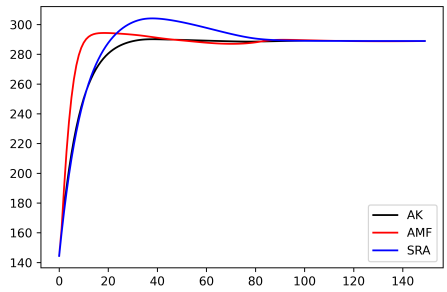
Table 4: Comparison of Three Solution Methods

	$J = 1$	$J = 2$	$J = 4$	$J = 8$
	Relative Execution Time			
AK method	1.000	1.930	3.681	8.025
	(54)	(50)	(48)	(51)
AMF method	0.784	1.566	3.158	6.398
SRA method	0.091	0.179	0.360	0.720
	(4)	(5)	(5)	(5)
	MAPD			
AMF method	1.87	1.88	1.83	1.67
SRA method	1.45	1.52	0.92	0.76

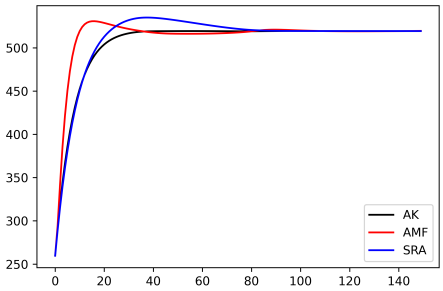
Figure 2: Comparison of Paths for Aggregate Capital



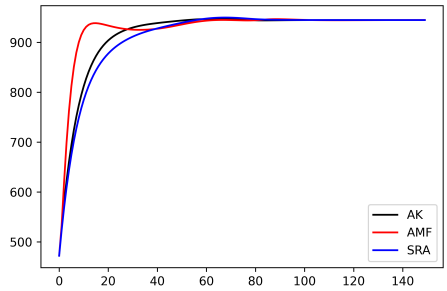
(a) $J = 1$



(b) $J = 2$



(c) $J = 4$



(d) $J = 8$

One way to leverage the gains from these faster methods without sacrificing accuracy would be to us the solutions from the AMF and SRA methods as starting values for the AK method. This would cut down on the number of time-consuming iterations that AK must perform. Table 5 shows the relative compute times and number of iterations for the AK method along with hybrid methods that use the starting values from AMF and SRA. The numbers for AK are identical to those in Table 4. For SRA the number of iterations is the total from the SRA preliminary run and the AK run. Since AMF has only one iteration, the number listed there is for the AK run only. The AMF hybrid now takes longer than simply running AK; on average 60% longer. In contrast, SRA runs an average of 18.2% faster than AK. The table

shows that the source of these longer compute times is due to the large number of iterations that must be run even after getting good starting values. For Table 5 the convergence criterion for the AK method was set to 1.0e-6. We would require fewer iterations with a less strict criterion. Accordingly, Table 6 replicates Table 5 with a larger criterion of 1.0e-2. The MAPDs with this new criterion are identical to those in Table 4 to three significant digits. The execution times for AMF are the same as before, but since AK runs faster, the relative times have increased. AMF is now slower than AK by an average of 49%, and averages 2.2 times as slow when we use the hybrid method. The news is better for SRA using the hybrid method it is now an average of one-third faster than AK. Intuition suggests that an even weaker criterion could yield even greater speedups without a large loss in accuracy, since the MAPD in the AK time paths across these two different criteria is less that 1.0e-7 for all values of J.

Table 5: Comparison of Relative Execution Times for AK with Two Hybrid Solutions, criterion 1.0e-6

	$J = 1$	$J = 2$	$J = 4$	$J = 8$
	Relative Execution Time			
AK method	1.000	1.930	3.681	8.025
	(54)	(50)	(48)	(51)
AMF hybrid	1.590	3.301	6.033	12.835
	(43)	(39)	(38)	(42)
SRA hybrid	0.091	0.179	0.360	0.720
	(45)	(43)	(40)	(42)

Table 6: Comparison of Relative Execution Times for AK with Four Other Solutions, criterion 1.0e-2

	$J = 1$	$J = 2$	$J = 4$	$J = 8$
	Relative Execution Time			
AK method	1.000	1.795	3.354	7.335
	(31)	(28)	(26)	(28)
AMF method	1.346	2.689	5.426	10.989
SRA method	0.098	0.197	0.389	1.005
	(3)	(3)	(3)	(4)
AMF hybrid	1.978	3.977	8.174	16.994
	(19)	(20)	(20)	(22)
SRA hybrid	0.629	1.171	2.358	5.288
	(19)	(18)	(18)	(20)

References

Auerbach, A. J. and L. J. Kotlikoff, *Dynamic Fiscal Policy*, Cambridge University Press, 1987.

Evans, Richard W. and Kerk Phillips, "OLG Life Cycle Models: New-Solution Methods and Applications," *Computational Economics*, 2014, *43* (1), 105–131.

Kotlikoff, Laurence J., "The A-K Model – Its Past, Present, and Future," Working Paper 6684, National Burea of Economic Research August 1998.

Miller, Merton and Charles Upton, *Macroeconomics: A Neoclassical Introduction*, Homewood, IL: R.D. Irwin, 1974.

Rausch, Sebastian and Thomas F. Rutherford, "Computation of Equilibria in OLGModels withMany Heterogeneous Households," working paper 2009.

Seidman, Laurence S., "Taxes in a Life Cycle Model with Bequests and Inheritances," *American Economic Review*, June 1983, *73* (3).

Summers, Lawrence H., "Capital Taxation and Accumulation in a Life Cycle Growth Model," *American Economic Review*, 1981, *71* (4), 533–544.

APPENDICES

The algorithm described in step (c) of Appendix A is used in all three methods described below and should therefore be coded as a stand-alone function.

A Algorithm for AK Method

1. Find the steady state.

2. Determine the initial state (perhaps a steady state, perhaps not).

3. Begin iterations over the whole time path by setting $i = 1$.

   (a) Assume a transition path for the aggregate capital stock, $\mathbf{K}^i = \{K_t^i\}_{t=1}^T$ and the aggregate labor supply, $\mathbf{L}^i = \{L_t^i\}_{t=1}^T$. K_1 is given by the initial state, $K_T = \bar{K}$, and $L_T = \bar{L}$.

   (b) Equations (2.5) and (2.6) give a path for wages and interest rates, $\mathbf{r}^i = \{r_t^i\}_{t=1}^T$, $\mathbf{w}^i = \{w_t^i\}_{t=1}^T$.

   (c) Given these factor price time paths, solve for the optimal consumption, labor supply, and savings for all households.

      i. Using the initial state to get $k_{s,j,1}$ for households alive in period 1, and $k_{1,j,t}$ for households born after the first period, make an initial guess for the the household's consumption in its first period. Equation (2.3) implicitly gives labor in that same period. Equation (2.2) gives consumption in the next period and (2.1) gives the capital carried to the next period. Chain this series of equations to get a life-cycle set of consumptions, labor supplies, and capital holdings, $\mathbf{c}_{sjt}, \mathbf{l}_{sjt}, \mathbf{k}_{sjt}$, noting that the last series runs through age $S + 1$.

      ii. There are three three cases when optimizing.

- Those who are alive in the initial period, $t = 1$, ages $s = 1$ to S

  Given final capital ($k_{S+1,j,S-s} = 0$), numerically search over the final consumption value (c_{Sj0} to find the unique value that sets capital in the first period ($k_{s,j,1}$) to that for the initial state.

- Those who are born after the initial period and die before the steady state, all ages born from periods $t = 2$ to $T - S - 1$.

  Given final capital ($k_{S+1,j,t} = 0$), numerically search over consumption value in the last period of life (c_{Sjt} to find the unique value that sets capital at the beginning of the lifecycle ($k_{1,j,t+S}$) to zero for that household.

- Those who will die on or after the steady state arrives all ages born from $t = T - S$ to $T - 1$.

  Given final capital ($k_{t-T,j,T} = \bar{k}_{t-T,j}$), numerically search over consumption value in the last period ($c_{Sj,T-1}$ to find the unique value that sets capital at the beginning of the lifecycle ($k_{1j,T-t}$) to zero.

iii. In all cases, save the corresponding $\mathbf{c}_{sjt}, \mathbf{l}_{sjt}, \mathbf{k}_{sjt}$ series.

(d) Use equations (2.10) and (2.9) to aggregate up the capital and labor from all households to generate a new set of capital and labor time paths, $\hat{\mathbf{K}}^i = \{\hat{K}^i_t\}_{t=1}^T$ and $\hat{\mathbf{L}}^i = \{\hat{L}^i_t\}_{t=1}^T$.

(e) Check for convergence by testing $|\hat{\mathbf{K}}^i - \mathbf{K}^i, \hat{\mathbf{L}}^i - \mathbf{L}^i| < \varepsilon$. If the paths have converged, cease iterating and use the calculated time paths for all variables as the solution.

(f) If the paths have not converged, construct a convex combination

of the original and new time paths.

$$\mathbf{K}^{i+1} = \rho\hat{\mathbf{K}}^{i} + (1-\rho)\mathbf{K}^{i}$$
$$\mathbf{L}^{i+1} = \rho\hat{\mathbf{L}}^{i} + (1-\rho)\mathbf{L}^{i}$$

(g) Update the iteration counter, i, and return to step (b).

In our Python code, we use an adaptive dampener that increases ρ if the distance metric has fallen relative to its value in the previous iteration; and decreases it if the distance has risen. We use the following updating formula.

$$\rho^{i+1} = \begin{cases} .5\rho^{i} & \text{if distance has risen} \\ 1-(1-\rho^{i}).95 & \text{otherwise} \end{cases} \tag{A.1}$$

We start with an initial value of $\rho^1 = .2$, and this algorithm causes the dampener to fluctuate mostly in the range of .15 to .4.

B Algorithm for AMF Method

1. Find the steady state.
2. Determine the initial state (perhaps a steady state, perhaps not).
3. Construct a simple alternative model to forecast the time paths of aggregate capital and labor. For example, a linear trend from the current value to the final steady state would give the following alternative.

$$K^{f}_{t+1} = K^{f}_{t} + \frac{\bar{K} - K^{f}_{t}}{T-t}$$
$$L^{f}_{t+1} = L^{f}_{t} + \frac{\bar{L} - L^{f}_{t}}{T-t}$$

4. Begin iterations over the periods in time path by setting $t = 1$. Assign the values of K_t and L_t from the initial state.

   (a) For every household alive in period t, use the alternative model to forecast K_t^f and L_t^f for the next S periods.

   (b) Equations (2.5) and (2.6) give forecast values for r_t^f and w_t^f for the next S periods.

   (c) Using the same procedure outline in the AK method step (c), solve for the planned time series $\mathbf{c}_{sjt}^p, \mathbf{l}_{sjt}^p, \mathbf{k}_{sjt}^p$.

   (d) Save the values of c_{sjt}^p, ℓ_{sjt}^p, and $k_{s+1,j,t+1}^p$ for each household alive in period t.

   (e) Using (2.10) and (2.9) aggregate to get K_{t+1} and L_{t+1}. Assign these as initial values for period $t + 1$.

   (f) If $t = T$ end the iteration, otherwise increment the time period, t, by one and move back to step (a).

C Algorithm for SRA Method

1. Set the iteration counter to $i = 1$.

2. Find the steady state for the RA model. Given benchmark data, calibrate the values for β^i and χ^i to match the share of aggregate consumption in GDP, and aggregate labor supply in total available time endowment for the steady state.

3. Determine the initial state (perhaps a steady state, perhaps not).

   (a) Solve the RA model starting at the initial state running forward for T periods, where T is large enough to approximately converge to the steady state.

(b) Using the values for interest rates and wages ($\mathbf{r}^i = \{r_t^i\}_{t=1}^T$, $\mathbf{w}^i = \{w_t^i\}_{t=1}^T$.) from this simulation solve for the optimal values of consumption, labor, and capital holdings ($\mathbf{c}_{sjt}, \mathbf{l}_{sjt}, \mathbf{k}_{sjt}$) for all households in the OLG model as in the AK method step (c).

(c) Recalibrate the RA model parameters β^{i+1} and χ^{i+1} to match values for the share of aggregate consumption in GDP, and aggregate labor supply in total available time endowment from the OLG model simulation above in period T, which is the OLG model's steady state.

(d) Check for convergence by testing $|\mathbf{r}^i - \mathbf{r}^{i-1}, \mathbf{w}^i - \mathbf{w}^{i-1}| < \varepsilon$. If the paths have converged, cease iterating and use the calculated time paths for all variables as the solution.

(e) If the paths have not converged, increment the iteration counter, i, by one and return to step (a). We find that the choice of the initial guesses for the transition paths of K_t and L_t make little difference in the time to converge. For simplicity we use as initial guess that these values are at their steady state levels for the whole simulation.

D Key Equations

Solving (2.2) for c_{sjt}.

$$c_{sjt} = \left(\beta[1 + (r_{t+1} - \delta)(1 - \tau)]\right)^{\frac{-1}{\gamma}} c_{s+1,j,t+1} \tag{D.1}$$

Solving (3.3) for ℓ_{sjt}.

$$\ell_{sjt} = \left[\frac{(1 - \tau)w_t a_{sj}}{\chi c_{sjt}^{\gamma}}\right]^{\frac{1}{\theta}} \tag{D.2}$$

Solving (2.1) for k_{sjt}.

$$k_{sjt} = \frac{k_{s_1,j,t+1} + c_{sjt} + f_t - (1-\tau)w_t a_{sj}\ell_{sjt}}{1 + (1-\tau)(r_t - \delta)} \tag{D.3}$$

E Finding the Steady States

E.1 OLG Model

The steady state versions of equations (D.1) - (D.3) are:

$$\bar{c}_{s+1,j} = (\beta[1 + (\bar{r} - \delta)(1-\tau)])^{\frac{1}{\gamma}}\, \bar{c}_{sj} \tag{E.1}$$

$$\bar{\ell}_{sj} = \left[\frac{(1-\tau)\bar{w}a_{sj}}{\chi \bar{c}_{sj}^{\gamma}}\right]^{\frac{1}{\theta}} \tag{E.2}$$

$$\bar{k}_{s+1,j} = \bar{w}a_{sj}\bar{\ell}_{sj} + (1 + r_t - \delta)\bar{k}_{sj} - \bar{c}_{sj} \tag{E.3}$$

We solve for the steady state by searching over values for $\{\bar{c}_{1j}\}_{j=1}^{J}, \bar{r}$ and $\bar{w}$ to satisfy the following conditions:

$$\bar{k}_{S+1,j} = 0; \forall j \tag{E.4}$$

$$\bar{r} = \alpha\frac{\bar{Y}}{\bar{K}} \tag{E.5}$$

$$\bar{w} = (1-\alpha)\frac{\bar{Y}}{\bar{L}} \tag{E.6}$$

With values for $\{\bar{c}_{1j}\}_{j=1}^{J}, \bar{r}$ and $\bar{w}$ and noting that $\bar{k}_{1j} = 0$, equations (E.1) - (E.3) allow us to iteratively recover $\{\bar{c}_{sj}\}_{s=2}^{S}$, $\{\bar{\ell}_{sj}\}_{s=1}^{S}$, and $\{\bar{k}_{sj}\}_{s=2}^{S+1}$.

Aggregating over s and j gives us $\bar{K}, \bar{L}$ and $\bar{C}$.

The remaining model equations give $\bar{Y}, \bar{I}, \bar{F}$ and $\bar{f}_{sj}$.

This allows us to evaluate (E.4) - (E.6).

E.2 RA model

The steady state version of (3.2) gives:

$$\bar{r} = \frac{\frac{1}{\beta} - 1}{1 - \tau} + \delta$$

We then search over values of $\bar{L}$ that satisfy:

$$\bar{C}^{-\gamma}(1 - \tau)\bar{w} = \chi \bar{L}^{\theta} \tag{E.7}$$

The steady state versions of (2.5) and (2.4) give:

$$\bar{K} = \left(\frac{\alpha}{\bar{r}}\right)^{\frac{1}{1-\alpha}} \bar{L}$$

Steady state versions of the rest of the model equations give us $\bar{Y}, \bar{w}, \bar{F}, \bar{C}$, and $\bar{I}$ allowing us to evaluate (E.7).