Lifecycle Modelling for Retirement in Julia (WORK IN PROGRESS)

Jordan Van, UNSW

2019
February

Abstract

In this paper, I demonstrate and walk through the implementation of various retirement lifecycle models in Julia. The underlying numerical theory is discussed and illustrated through code excerpts with reference to features of the lifecycle models. In total, four models of increasing complexity are presented. We examine models of optimal consumption, annuitisation and insurance and explore the challenges associated with the addition of each new feature. Finally, I illustrate the capabilities of the models in predicting individual behaviour in different circumstances under a rational framework. The model's code has been made freely available, and the reader is encouraged to try the code for themselves.

Contents

1 Introduction

1.1 Status of this report

This report is currently unfinished. However, I hope that you continue to read on.

Additionally, the code is also in a fairly messy state; for every two helpful comments, there is a residual note-to-self or a commented-out line of code used for testing. Don't get me wrong, I'm sure that every bit of useful information about the code has been included in the comments, but the overabundance of comments makes it difficult to distinguish between important information and throwaway remarks. With the eventual completion of this report, I will look to clean up the code such that it's suitably documented for readers of this report.

1.2 Introduction

Lifecycle modelling is a fundamental tool in modelling individual financial decisions and predicting behaviour under different scenarios. This paper tackles lifecycle modelling in the context of retirement, however, the numerical and computational techniques are not specific to retirement. Although the exact techniques vary between each model, they generally fall under the umbrella of numerical dynamic programming.

For simplicity sake, we number the models 1-4. The models were created in that order, but that is not to imply that they are necessarily direct extensions upon their preceding model. Model 2 and 4 are built on Model 1, whereas Model 3 extends on Models 2s features but was programmed separately. The code is available at

`https://github.com/JordanVan/LifeCycleModelling`.

Model 1 introduces a savings and consumption model for an individual retiree. During each period, the individual must decide how much to consume, and consequently how much to save. The savings are carried over to the next period where they accumulate a fixed interest rate. Entering each period, the individual also has the possibility of dying. Each period, the individual receives utility from consumption subject to temporal discounting. Cognizant of these factors, the individual seeks to maximise their total utility over their lifecycle.

Model 2 introduces the option to purchase an annuity during the first year of retirement. The pricing of the annuity is actuarially fair, meaning that the price is equal to the expected present value. The annuitisation of the individual is a new state variable which means that entering a new period, the individuals cash on hand is now a function of both their savings from the previous period and the amount of annuitisation they are entitled to. The annuitisation level does not change after the first period, allowing us to treat this new state variable as a parameter during computation.

Model 3 takes the annuitisation offered in Model 2 but allows for it to be purchased in any time period. As such, the coupon paid per unit premium depends on the age at which it was purchased. The option to annuitise at any time also introduces the challenge of an additional control and state variable. As such, a different dynamic programming technique was used compared to the other models.

Model 4 introduces health insurance to Model 2. The individual now has a chance of contracting an illness which has an associated health cost and increased mortality rate. Two types of illness are introduced, critical illness, which is casually referred to as cancer, and long term care(LTC) illness. The former carries a large one off cost and sharp increase in mortality, while the latter has a lower but sustained cost and a moderately increased mortality. Like the annuity policy, the

insurance is actuarially fair in its pricing.

The code is written in Julia, a high-level scientific programming language, whose features and speed have recently attracted many economists. Readers familiar with Python will find the transition to Julia smooth as the syntax is largely similar but should refer to the official documentation to ensure no misunderstandings occur. Although the code written and presented is entirely original, it makes use of external open source Julia modules. Furthermore, I also acknowledge that the implementations of the endogenous grid method in Models 1,2,4 are based off an earlier personal Julia rewrite of the MATLAB code presented in Caroll (2006) and that the 2D implementation of the Bellman operator in Model 3 was written with reference to the 1D implementation featured on QuantEcon(Sargent and Stachurski, 2017).

2 1 - Consumption/Savings

2.1 Overview

I will begin with a description of the problem including the parameters, and mechanics of the lifecycle, followed by the theory behind the solution and methods for practical implementation. Model 1 is straight forward in both theory and implementation. However, the large number of new concepts may be confusing to readers who are new to numerical dynamic programming, so those readers are suggested to ensure they understand the concepts before continuing as this model will serve as the foundation for the upcoming models. This model has one control variable, consumption, and one state variable, savings, excluding time. Notable code and aspects of the model can be found in Appendix A1.

2.2 Theory and Numerical Methods

In Model 1, we begin modelling an individual's life from $retirementAge$ to $retirementAge + PeriodsToAdd$. $retirementAge$ is defined such that the final age in life, $retirementAge + PeriodsToAdd$ is always equal to 105, which is taken to be the final period of life. The individual begins with $initialWealth$ and will receive no more income for the rest of their life. Entering period t, the individual begins with x_t cash on hand, determined by their choices in previous periods, and chooses how much to consume c_t and save s_t, subject to a liquidity constraint preventing overspending $c_t + s_t = x_t$. Consumption yields utility according to the CRRA utility function $U(c) = \frac{c^{1-\rho}}{1-\rho}$ with risk aversion parameter ρ, subject to intertemporal discount factor β. The savings s_t are carried into next period and accumulate interest at gross interest rate R. Mortality has also been programmed into the model. The chance of surviving into period t from age $t-1$ is given by $\phi(t)$. Thus, the individual has the goal of choosing policy function $c_t(x_t)$ to maximise their expected total utility without overspending i.e.

$$\begin{aligned} \underset{c_t(x_t)}{\text{maximise}} \quad & E\left[\sum_{n=0}^{T} \beta^t u\left(c_t\left(x_t\right)\right)\right] \\ \text{subject to} \quad & x_t = R\left(x_{t-1} - c_{t-1}\right), \\ & x_0 = initialWealth \end{aligned}$$

The complexity of the problems makes any attempts to analytically solve the problem for more than two periods near impossible. Instead, we solve the problem numerically using the endogenous grid method(EGM), a numerical dynamic programming technique first introduced in Caroll (2006). The EGM is a method for solving the Euler Equation whose exact implementation is dependent on the utility function used in the model. The Euler Equation is a necessary condition for the consumption policy to be optimal. Intuitively speaking, if a consumption policy is optimal(maximises expected utility), there should be no gain from saving a marginal unit in a period to instead consume in the following period. Note that in the following period, we must accrue interest, discount for time and account for

the chance of death on that saved marginal unit. Equating the marginal expected utilities, we obtain the Euler Equation

$$U^{'}(c_t(x_t))) = \beta R \phi(t) U^{'}(c_{t+1}(R(x_t - c_t(x_t)))) \text{ for } t = 1..PeriodsToAdd \tag{1}$$

Note that we do not need to take the expectation of the next periods marginal utility because under this model, the next period cash on hand is deterministic. The chance of death, while stochastic, has no bearing on the next period's cash on hand and can be accounted for simply through the term $\phi(t)$. For feasible(positive) consumption levels, the derivative of the consumption function is invertible. The inverse of the CRRA utility function is $c^{-1/\rho}$. Inverting both sides of (1), we obtain

$$c_t(x_t) = (\beta R \phi(t) U^{'}(c_{t+1}(R(x_t - c_t(x_t)))))^{-1/\rho} \text{ for } t = 1..PeriodsToAdd \tag{2}$$

The observant reader will notice that c_t appears on both sides of the equation and as such, $c_t(x_t)$ can not be easily determined without solving this functional equation. The ingenuity (and speed) of the endogenous grid method comes from defining an **exogenous** grid of savings s_t from which x_t can be **endgenously** found via $x_t = c_t + s_t$. So, given a savings level s_t, one can find the associated level of optimum consumption with the equation

$$c_t(s_t) = (\beta R \phi(t) U^{'}(c_{t+1}(R(s_t))))^{-1/\rho} \text{ for } t = 1..PeriodsToAdd \tag{3}$$

from which cash on hand can be reconstructed, using $x_t = c_t + s_t$, to find $c_t(x_t)$. This can be repeated for every saving grid point, of which there are n, to construct a grid of cash on hand points. The value of the saving grid points are not dependent on t, but the notation $(s_{i,t})_{i=1..n}$ is adopted because the reconstructed cash on hand points $(x_{i,t})_{i=1..n}$ depend on $(c_{i,t})_{i=1..n}$, and hence vary with t. We also require in advance the next period consumption function c_{t+1}, which we will ignore for now.

The reason why we choose to solve over a grid of saving points instead of attempting to solve over the infinite possible values of s_t is obvious. However, it is guaranteed that at some point solving the life-cycle model, we will have to query

the consumption for a cash on hand value which is not on our grid. To deal with this, we use interpolation, a method of estimating the value for points outside our grid points. There are many methods for interpolating data but we stick with the simplest, linear interpolation, which you can imagine as simply drawing straight lines between the data points. This also has the nice property of preserving the concavity and monoticity of the true function. My experiences with using higher dimensional splines (quadratic and cubic) have almost always resulted in unstable results, so I advise any reader concerned with accuracy to increase the number of grid points (n, as it appears in the code). With this in mind, we can view storing the optimal consumption **values** on these grid points, $c_t(x_{i,t})_{i=1..n}$, as a way of efficiently storing an approximation of the optimal consumption **function** $c_t(x_t)$.

The last thing we need before finding the optimal consumption policy for a time period t is that optimal consumption policy for time $t+1$. This can be assured by solving the model backwards in time from the final period, a form of backwards induction. By setting age 105 to be the final period of life, the optimal consumption in this period is to simply consume all cash on hand. From this we work backwards, storing the optimal consumption function at each period as an array of values containing the consumption at different grid points and reconstructing it via interpolation as needed. We do not cover the case where an individual possesses a bequest motive, but the principles are nevertheless the same. We previously mentioned the need for a liquidity constraint to prevent overconsumption; as it turns out, Model 1 performs identically with and without a liquidity constraint being implemented. As a result, we will leave the discussion of implementing a liquidity constraint for later models.T his concludes the endogenous grid method. We will now present some psuedocode of the method, and discuss methods of representing the solved results with figures.

2.3 Psuedocode

2.3.1 Solving for the Optimum Policy

```
/* Defining the savings grid and initialising optimal final
   period consumption.  In practice, we do not use an evenly
   spaced savings grid.  See appendix A1 for more details    */
savingsGrid = [0,1,..,n-1] * 100/n
consumption = [0,1,..,n-1] * 100/n
cashOnHand = [0,1,..,n-1] * 100/n
/* We are solving the model backwards, starting from the
   penultimate period                                        */
for i in 1:PeriodToAdd do
    /* This is equation 3 for t = 104,103,...               */
    newConsumption = inverseUPrime(margSavValue(106-i,SavingsGrid)
    newCashOnHand = newConsumption + SavingsGrid
    /* We have a set of cash on hands(stored in the 1d array
       above), and a corresponding set of optimal consumption
       values(2 lines above) i.e.  the optimal policy for the
       (i+1)nth last period.  We now store(concatenate) this
       data to the 2D array which holds the complete policy  */
    concatenate(consumption,newConsumption)
    concatenate(cashOnHand,newCashOnHand)
end
```

We define some of the functions used above, namely "margSavValue" and its subfunctions. We avoid assigning an explicit function to "inverseUPrime" and remind the reader that this is dependent on the utility function chosen. However, we do note that using the CRRA utility function, we have $inverseUPrime(c) =$

$c^{-1/\rho}$ and "UPrime", which we will soon see, is given by $UPrime(c) = c.^{-\rho}$.

Function `margSavValue(`*age,savingsGrid*`):`

```
/* This is the marginal value of saving a unit at age-1 to
   spend at age.  R is the gross interest rate            */
```

nextPeriodCash = R*savingsGrid

expectedUPrime = $\beta\phi$(age)R*UPrime(nextPeriodCon(nextPeriodCash))

```
/* Note that R appears twice.                             */
```

return expectedUPrime

Function `nextPeriodCon(`*nextPeriodCash*`):`

```
/* Although it does not take next period t+1 as a parameter,
   this function DOES depend on it.  However, taking the end
   of the policy data will always give us the policy from
   t+1, so there is no need for a time input.             */
```

Xdata = X[:,end]

Cdata = C[:,end]

interpolatedConsumption = interpolate(Xdata,Cdata,type = linear)

consumptionValues = interpolatedConsumption(nextPeriodCash)

return consumptionValues

Although solving for the optimal policy is nice, it is absolutely vital that there is an easy way to verify the validity of these results. We now walk through some ways in which the results can be presented graphically and mention briefly how one might check the validity of the results. Following this, we will include some of these visual representations in the next subsection.

2.3.2 Plotting results

The most obvious way to represent the solution to the model is to directly graph the optimal consumption functions for each time period. Recall that for each time period, we have a set of consumption and cash on hand pairs. These can be graphed with the time of each period being represented by the colour of the graph. A convenient feature of this model is given an initial starting wealth, an individual's behaviour in each period is perfectly predictable. Death is uncertain, but it immediately ends the life cycle so it can be effectively ignored. Consequently, we can graph both the optimal consumption and saving over time given an initial wealth by following through an individual's life as if they had lived to the final period. This graph can be interpreted as how much one would consume and save **if** they were to survive to that age. Below we present some psuedocode which produces the two aforementioned graphs.

```
/* TO BE COMPLETED...                                          */
```

2.3.3 Verifying Results

With a way to visually represent our results, we can now more easily perform some sanity checks on our model. The simplest sanity check would be to manually solve the model for 2 or 3 periods and compare the solved optimal policy with that of the numerical model. If the results are identical, then it likely that the model is valid for any number of periods. Following this, it is a good practice to play with parameters to see if the effect on results is as expected. For example, a lower intertemporal discount factor β should result in higher consumption during earlier periods of life. Experimenting should parameters shouldn't be limited to realistic scenarios as edge cases can provide insight into the inner workings of your model. As an example, massively increasing the mortality rate in at age 76 should produce a clear distinction in consumption levels before and after 76. Indeed, this is what we observe.

2.4 Figures

I lied, there are no figures. Well, there are actually some images if you check the folder on Github containing the code, but soon you won't have to! [Some comments about the figures.]

3 2 - Consumption/Savings/Initial Annuities

3.1 Overview

Model 2 introduces the opportunity for the individual to annuitise their wealth at retirement. They are allowed to annuitise any integer percent of their initial wealth(for computational simplicity). The annuity's are priced at an actuarially fair price, in the sense that the price is set to the expected total payout of the policy. The annuities pay every year until death, starting from the purchase year. In order to tackle this problem, we consider a fixed annuity level and its effects on the relevant equations to be solved. We then solve the model for every allowed annuity level, generating the optimal policy. Because the policy does not contain the optimal initial annuity choice, we then run simulations to find the optimal initial annuity purchase.

3.2 Theory and Numerical Methods

We continue on from Model 1 but modifying the Euler Equation previously presented (3) to account for an annuity payment. However, we first define a 2 dimen-

sional grid of saving and the annuity levels.

The annuity level is represented by the proportion of initial wealth that was spent on annuities which allows for convenient discussion. The 2 dimensional grid of saving and annuity levels $(s_{i,t}, a_{j,t})_{i=1..n,j=1..n}$ can be thought of as the Cartesian product between two seperately defined grids for the saving level $(s_{i,t})_{i=1..n}$, and the annuity level $(a_{j,t})_{j=1..n2}$. While n can be freely chosen depending on accuracy requirements, we suggest $n = 20$ as higher values of n do not produce noticeably more accurate results. On the other hand, $n2$ should be restricted to 101, which combined with an evenly spaced grid between 1 and 100, gives a grid of integer percentages. Higher levels of $n2$ are of course possible, but there is little to be gained from distinguishing whether optimal annuitisation is 73% or 73.5%, certainly not enough to justify a 2 time slow down.

3.3 Psuedocode

asdfadsfadsf

3.4 Figures

asdfadsf

4 3 - Consumption/Savings/Arbitrary Annuities

4.1 Overview

asdf

4.2 Theory and Numerical Methods

asdf

4.3 Psuedocode

asdf

4.4 Figures

asdf

5 4 - Consumption/Savings/Initial Annuities /Health Insurances

5.1 Overview

adsffads

5.2 Theory and Numerical Methods

asdfadfs

5.3 Psuedocode

asdfads

5.4 Figures

sdfg

6 Applications

sfdg

6.1 Overview

adsfads

6.2 Adverse Selection

asdfad

6.3 Government Pension

adsf

6.4 Delayed Annuitisation

agsgds

6.5 Price Sensitivity of Different Health Insurances

agsagds

7 Conclusion

In conclusion, I conclude.

Appendix A Notable Code and Methods

A.1 Model 1

Unevenly spaced points are more efficient.

A.2 Model 2

CAYG(Constrain As You Go) liquidity constraints.

A.3 Model 3

The penalty method lets impose constraints in spite of optimisation package used.

A.4 Model 4

Tensors are confusing.

References

Caroll, C. D., 2006. The method of endogenous gridpoints for solving dynamic stochastic optimization problems. Economic letters 91 (3), 312–320.

Sargent, T. J., Stachurski, J., 2017. Optimal growth i: The stochastic optimal growth model.
URL `https://lectures.quantecon.org/jl/optgrowth.html#Computation`