**Teacher and Student Outcomes Code Documentation**

This document details the solution method for the teacher and student outcomes model. The code follows the model in “Teacher Model2019-taber.pdf” very closely, with the exception that non-teaching advanced degrees are taken out for simplicity.

**Setup**

The code begins by initializing model primitives and state grids and reading in the current guess of parameter vectors and reshaping them into something more easy to use (function “Reshape\_param” in the tso\_utilities.jl file). The packaged model parameters are then initialized as a struct to allow for them to be passed easily between functions. The current version of the code makes the following state space reductions in order to maximize tractability – while this is not intended to be the final state space, this enables the code to be vetted quickly and gives us information about how long the code is likely to take with a richer specification.

* College quality, unobserved ability, and unobserved tastes for each occupation are set to be binary. We have two occupations, so this gives a total of 2x2x2 = 8 unobserved types.
* College major is assumed binary (not teaching and teaching). Number of occupations J is set to 2 (the same)
* I set T (number of post-schooling periods) = 10. This both substantially reduces the state space (since experience is multidimensional, the experience state space size increases exponentially with T) and is also motivated by the fact that the final B&B interviews for a cohort are 10-year follow-up surveys.
* Teacher quality (xi) is discretized to [-1. 0, 1]. In general, because teacher VA is assumed normal, I think a good approach will be to discretize it into N equal-mass points.

With model parameters and state grids established, we can begin actually solving the model. With the current parameterization that emphasizes tractability, the code solves all the value functions in approximately 16 seconds. The state space for Phase D is:

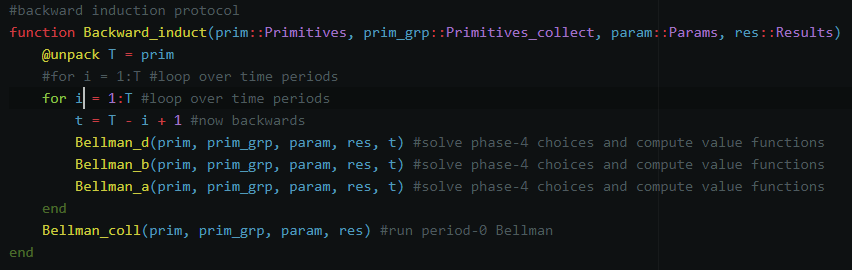
2 (college quality) x 2(gender) x 2(race) x 8(unobserved heterogeneity) x 2(major) x 2(advanced degree) x 2(license) x 3(teacher quality) x 2(ever taught) x 3(last-period occupation) x 2(teaching offer) x 261(experience states) x 10(periods) x 3(occupation selections) = **36,495,360**

As a note, functions that govern things like flow utility from occupations and costs from license/degree/major choices can be found in the tso\_background\_functions.jl file. These functions generally follow directly from the parameterization in “Teacher Model2019-taber.pdf” and are fairly straightforward, so they are not discussed in depth here.

**Model Solution**

The model is solved via backward induction. I start at period T = 10, and then work backwards over period 9, 8, 7 . . . until reaching period 1. For each period *t*, the code computes, in order:

* Period-*t* Vd value functions (value functions for occupation decisions)
* Period-*t* Vb value functions (value functions for license/certification decisions)
* Period-*t* Va value functions (value functions for advanced degree decisions)



*Backward induction code. Notes: the arguments passed are called structs, which are essentially composite types that can hold many individual variables. Prim holds model primitives and state grids. Prim\_grp holds a few composite states (X for observable characteristics, chi for unobserved heterogeneity, and multidimensional experience vector) for ease. Param contains model parameters to be estimated. Res contains model value functions.*

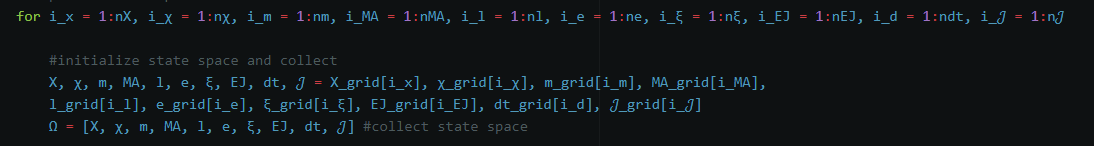
After computing the period-1 Va function, the model computes the period-0 major choice value function, at which point the model is solved. Value functions are computed for all possible combinations of state variables and choices net of the Gumbel shocks that influence each choice. When simulating data, these value functions will be used in conjunction with Gumbel draws to actually determine agent behavior.

I now turn to describing in depth how I solve for value functions in each sub-period.

**Vd Solution (Occupation Choice)**

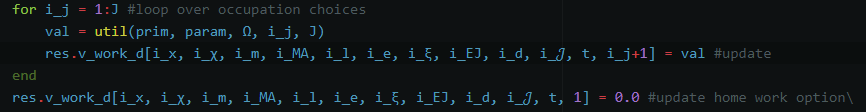
**Terminal Period**

The first thing the code does is check whether we are in the terminal period (T=10). If we are, things are simplified considerably. Whether or not we are in the terminal period, however, I begin by constructing a loop over the state space indices, translating these indices into state space values, and collecting these as a vector Omega that can be passed easily to other functions.



In the terminal period, I then loop over occupation choices, compute flow utility, and store the results. Two things to note:

* Whether the individual has a teaching job offer in the current period (script J) does not influence flow utility from any occupation in the current period, so we don’t have to worry about it here. The teaching offer probabilities, rather, will influence continuation values in the Vb functions..
* If the individual chooses to work as a teacher for the first time in the terminal period, we don’t have to worry about the value-added learning procedure. The agent has no private information about their quality, and their quality is mean-zero, so it will fall out of the expectation. The util function is written so that the agent does not get any utility from their individual quality if their EJ (dummy for ever having taught) is zero.



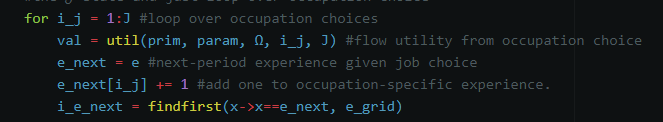
*Note that the language I am writing in does not have 0-indexing, so the non-teaching choice occupies index 1 in the final dimension here. Recall that the value of not working here is normalized to zero.*

**Non-Terminal Period**

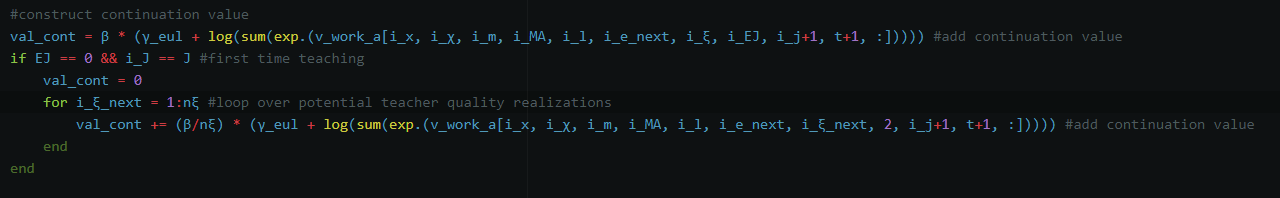
Conceptually, solving the phase-d value function in a non-terminal period proceeds similarly, except that now we need to keep track of three new things:

* (potential) learning about one’s own teaching quality
* Evolution of experience and job-worked-last-year (dt-1) in the model.
* Whether one already has an advanced degree or not: this will influence whether the agent has any choice to make in phase A of the next period.

As before, I begin by defining state space indices and values and looping over occupation choices. For each choice, I first compute flow utility and then figure out where in the experience state space grid (in terms of index) the choice will take us to:

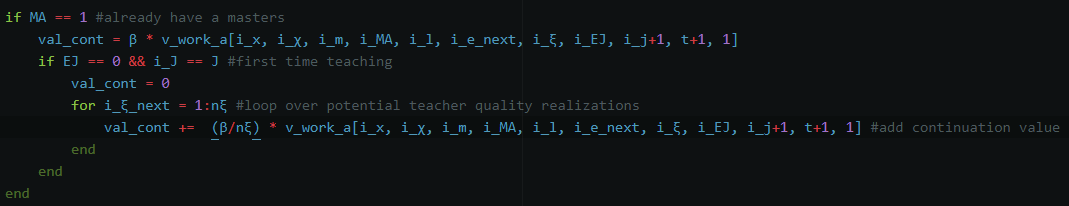


Next is to construct continuation values. I first do this for the case for the individual does not have an advanced degree. The continuation value here is given by the usual logit formula. I then check whether the individual is selecting the teaching occupation for the first time. If this is the case, then the continuation value is constructed as an equally-weighted (recall the assumption of an equal-mass discretization of xi) average of the next-period Phase A value function over possible teacher quality levels:



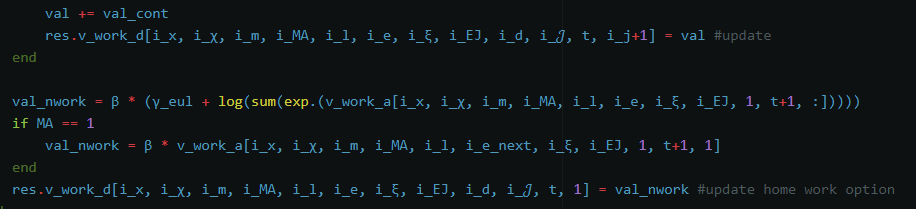
*Notes: gamma\_eul is the Euler-Mascheroni constant. The i\_j+1 in the vectors reflects the state corresponding to job-last-worked updating. Note that allowing for stochastic learning of one’s teaching ability (e.g. Poisson arrival of teacher quality information) would be trivial to implement here.*

Next, I handle the case where the agent already has a masters. In this case, the agent has no choice to make in the upcoming Phase A and knows which state they will map into with certainty. **Question to Chao/Chris: Since there’s no advanced degree choice to make, does that mean the agent no longer receives the utility shocks that govern advanced degree acquisition? If so, it seems this would reduce the continuation value in the preceding Phase D.**



*Note: if the agent already has an MA, the phase-A value function is equal across all indices in the final dimension (the final dimension is the dimension over degree acquisition choice, which is redundant if the agent already has one). In other words, the “1” in the final part of the vector isn’t significant.*

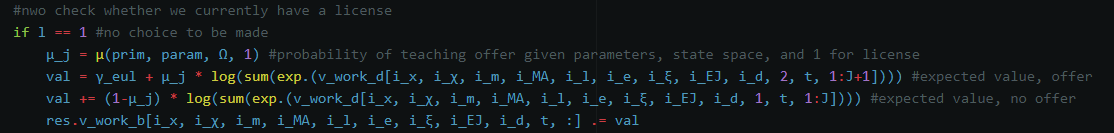
I then add the continuation value to the flow utility and update the value function before doing the same thing for the non-working option.



*Note: The non-working option has zero flow utility but still gets the appropriate continuation value.*

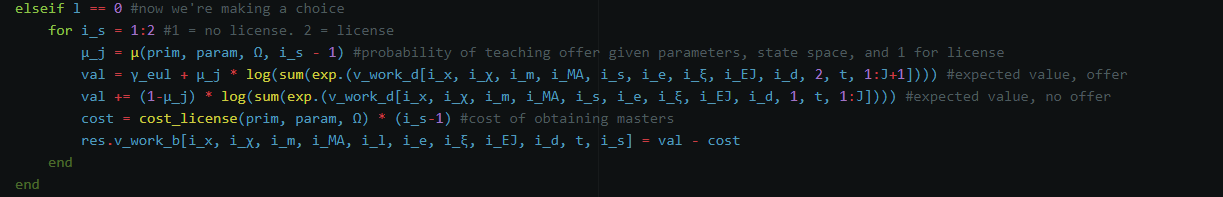
**Vb Solution (License/Certification Choice)**

As per usual, I begin by looping over the state space indices and storing state space values. I then check whether the agent already has a license. If so, I compute their likelihood of a teaching offer (mu) and the resultant continuation payoff. The teaching offer in influences the continuation payoff through altering how many choices the agent will have in the upcoming Phase D, which is captured easily in the usual logit formula:



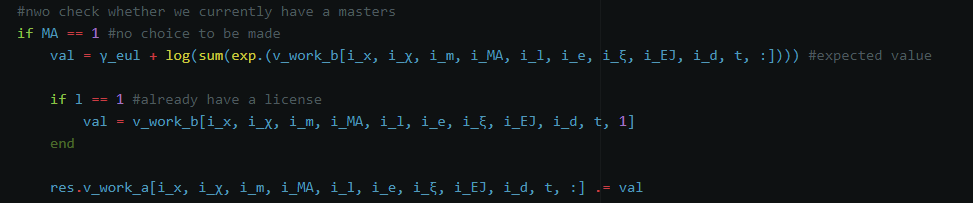
*Note: the mu function includes a cubic polynomial in teaching experience a dummy for having a teaching license. Both of these are then interacted with whether the individual taught in the previous period (which also has its own main effect).*

If the individual does not already have a license, then I loop over license choices (don’t get one, get one). I compute the probability of a teaching offer in the upcoming Phase D given either case, which influences the continuation payoff, and I include the cost of obtaining a license should the agent choose to do so:

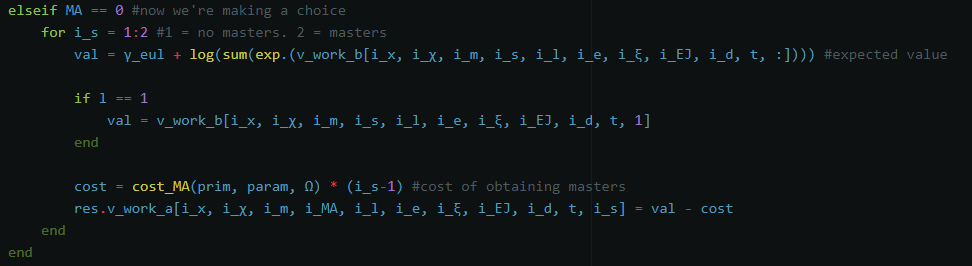


**Va Solution (Advanced Degree Choice)**

As per usual, I begin by looping over the state space indices and storing state space values. I then check whether the agent already has an advanced degree. If so, then I simply compute the continuation value associated with the upcoming Phase B (following the usual logit formula). Note that here I also check whether the agent already has a **license**, in which case the agent will have no decision to make in the upcoming phase B and will have their continuation value adjusted similar to the adjustment we made for agents who already had an advanced degree in Phase D.

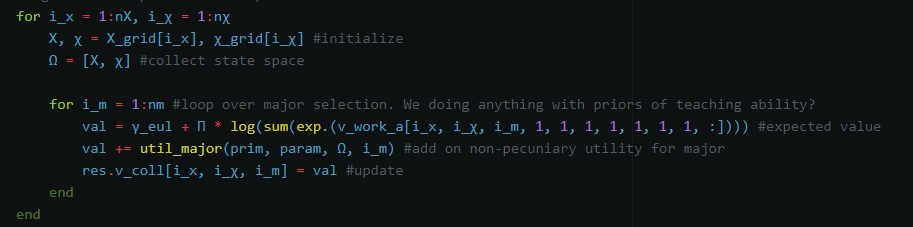


If the agent does not already have an advanced degree, then I loop over possible advanced degree selections (don’t get one, get one), after which I compute continuation values as and factor in costs to obtaining an advanced degree should the agent choose to do so:



**V0Solution (Major Choice)**

After having solved backwards through the period-1 Phase A value function, calculating the period-0 value function is as simple as looping over demographics and unobserved types, after which I loop over major choices and compute continuation values and flow utilities associated with major selections:



After this, the model is solved.