Household Problem and Distributions

In Nygaard, Sorensen and Wang (2020), we study the optimal allocation of COVID-19 welfare checks. Congress spent $250 billion sending checks to individuals in March 2020 to provide economic stimulus. Could the same amount of stimulus have been achieved for less money? Using a life-cycle consumption-saving model with heterogeneous consumers, we calculate the consumption responses to cash transfers for, e.g., couples and singles with different levels of income and number of children. We calculate the aggregate consumption response for all feasible allocations of $250 billion and, using a new algorithm that allows for the ranking of an arbitrarily large number of allocations, we find the optimal allocation under alternative constraints. The optimal policy allocates more toward low-income and younger consumers and can achieve the same stimulus effect at almost half the cost.

This Matlab based programming guide, package, and associated vignettes, provide examples and instructions on how the dynamic programming problem in Nygaard, Sorensen and Wang (2020) is solved. The R optimal allocation package PrjOptiAlloc takes inputs from the dynamic programming problems and solves for optimal allocations given varying planner objectives and constraints.

Flat Script and Code Package

There are two broad versions of the code. A number of files are included in the zflat folder, including the operation gateway file main. Files in the zflat folder provides a linear, easier to understand illustration/ demonstration of the overall code structure. It is useful to review the overall algorithm design. However, it should not be called to implement the programs. Programs in the folder were written to help test out algorithm ideas.

The rest of the files inside PrjOptiSNW form a matlab package that can be downloaded and installed. Each component of the overall code program is programmed up separately with its own testing vignette and default parameter structure. Various solution algorithms are provided at each step, with the final checks problem relying on efficient and precise solution methods.

Dynamic Programming Solution Structure COVIDless World

First we solve for the optimal consumption/savings problem in the COVID-less world:

- **83**: 2020 age groups, age 18 to 100 age groups
- **65**: grid of savings state-space grid, and exact continuous optimal savings choices using the **FF_VFI_AZ_BISEC_VEC** function from **MEconTools**.
- **6650 shocks**: 1330 productivity shocks for household head and sposue and 5 kids transition count shocks
- **2** permanent education states
- **2** permanent marital states

The state-space has: 2*2*6650*65*83= 143,507,000 elements. The choice-space is is continuous. Two important things two note:

1. The large number of shocks are needed to obtain accurate group-specific marginal propensity effects for small income bins that define the choice-set of the allocation problem.

2. While a choice-grid-based solution algorithm might sufficiently approximate the value function, but its policy function zig-zags. For the welfare checks problem, where welfare checks come in small increments, the zig-zags lead to fluctuating (negative and positive) marginal propensities to consume as resource availability increases for very small amounts of check increments. To deal with this challenge, we rely on the **FF_VFI_AZ_BISEC_VEC** function from **MEconTools** to provides efficient exact savings choices.

Solving this dynamic life-cycle programming problem requires approximately 10 to 20 minutes on a home-pc depending on computer speed. There is no processor requirements. Memory requirement is approximately 20GB. There are two core associated functions vignettes that solves the dynamic programming problem to obtain value/policy and distributions induceds by exogenous processes and the policy function:

- Core dynamic programming code: snwx_vfi_bisec_vec
- Core distribution code: snwx_ds_bisec_vec

Small testing vignettes of alternative solution algorithms for policy/value:

- Small test using matlab minimizer (very slow but identical results as core program): snwx_vfi_test
- Small test using grid-search-based solution algorithm (insufficiently precise for welfare checks): snwx_vfi_test_grid_search
- Small test of core dynamic programming code: snwx_vfi_test_bisec_vec
- Small test of core dynamic programming code with spousal shock: snwx_vfi_test_bisec_vec_spousalshock

Testing vignettes for alternative solution algorithm for distribution:

- Grid serach distributional code (insufficiently precise) :snwx_ds_grid_search
- Core solution distribution code (vectorized for policy/value, looped for dist): snwx_ds_bisec_vec_loop
- Core solution distribution code (vectorized fully): snwx_ds_bisec_vec

Dynamic Programming Solution Structure during COVID Year

During the COVID year, we use the value function from the COVID-less world as the continuation value, and solve for consumption-savings policy/value functions during the COVID year. We solve once for households facing realized COVID surprise unemployment shocks, one more time for households who do not experience COVID unemployment shocks.

We solve for the marginal consumption differences and value given 244 increments of checks ($100) each check. This is done again by using the **FF_VFI_AZ_BISEC_VEC** function from **MEconTools**. While checks could be viewed as an additional state variable, we evaluate the marginal effects of check by solving for the equivalent household-specpfic variation in savings state that has the same effect as a welfare check transfer. The process takes into account the nonlinear tax-schedule that households face as well as return on savings.

Overall:

- 286 million: Solve 143 million state-space points twice under COVID unemployment and COVID employment world

- 70 billion: Solve at the 143 million state-space elements 244 + 1 times for all possible check levels (244 checks + no check value/consumption) to arrive at 70 billion marginal propensity to consume for households with heterogeneities in education level, marital status, children below 18 count (0 to 4), age, savings levels, household head and spouse shocks.

Associated functions vignettes: the core dynamic programming code: snwx_vfi_bisec_vec, has a third input which is the existing future value function. When this is provided, the dynamical programming problems solves for one period given already computed future value, and so the dynamic programming solution solves forward. When it is not provided, solves for value/policy backwards.

- snwx_vfi_unemp_bisec_vec provides the vignette given unemployment shock.
- snwx_a4chk_wrk_bisec_vec computes the marginal impacts of a particular welfare check increment for those without unemployment shock in COVID year.
- snwx_a4chk_unemp_bisec_vec computes the marginal impacts of a particular welfare check increment for those with unemployment shock in COVID year.
- snwx_evuvw20_jaeemk considers probabilities for getting hit with the COVID shock and considers the expected value conditional on age, savings level, shocks, educational status, kids count and marital status in 2020.