ECON 899: Problem Set 1

Katherine Kwok

September 2021

Overview: For this assignment, the goal is to add uncertainty over technology shocks to a dynamic programming problem in an infinite horizon growth model. In particular, we are asked to modify existing matlab *or* julia code, and fortran code. I have written up the code in julia, fortran, and a parallelized verion of the julia code.

Dynamic Programming Problem: The dynamic programming problem is as follows

$$V(k,z) = \max_{k'} \log(c) + \beta \sum_{z' \in Z} \pi(z'|z) V(k',z')$$
$$\text{s.t. } k' = (1-\delta)k + zk^{\theta} - c$$

where $z \in Z = \{Z^g, Z^b\}$, and $\pi(z'|z)$ is a given cell in the transition matrix

$$\Pi = \begin{pmatrix} 0.977 & 0.023 \\ 0.074 & 0.926 \end{pmatrix}$$

We are also given the parameters $\theta = 0.36, \delta = 0.025, \beta = 0.99$.

Value Function: Using julia and fortran, we can compute the value and policy functions that solve the problem. I find that the value function is increasing over k and concave. The plots of the value functions in both good states and bad states are below.

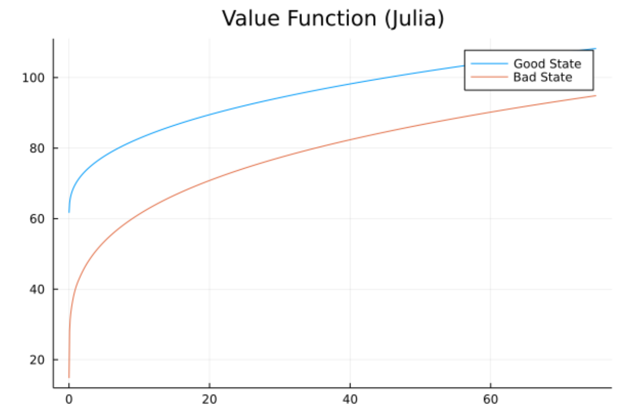

Figure 1: Value Function from Julia Code

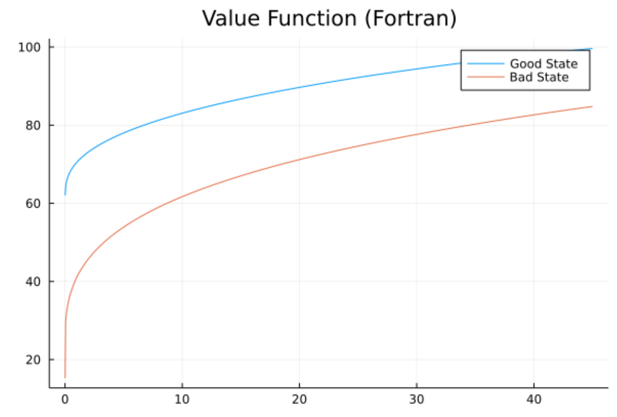

Figure 2: Value Function from Fortran Code

Policy Function and Savings: I have plotted the policy functions and the changes in the policy functions for both good and bad states below. Observing the plots below, we see that the policy function is increasing over k and z. The policy function slopes upwards, and the function for the good state is higher than that of the bad state.

To see how savings are affected by k and z, we plot the changes in the policy functions, i.e. $k'(k, z) - k$ across k for each possible state z. In the bad state, savings decrease as k increases. In the good state, savings initially increases with k, but curves downwards eventually.

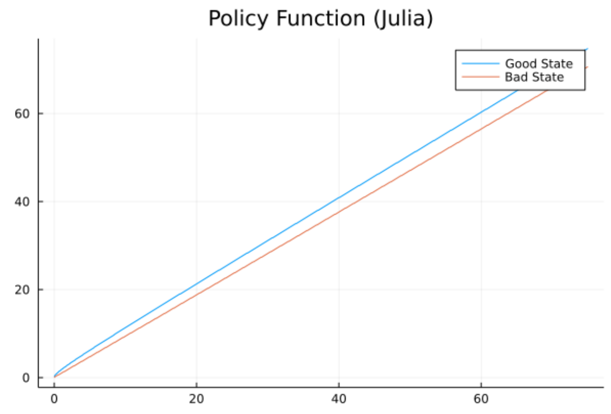

Figure 3: Policy Function from Julia Code

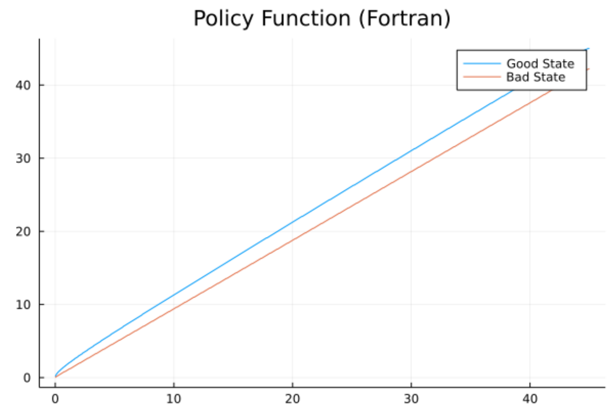

Figure 4: Policy Function from Fortran Code

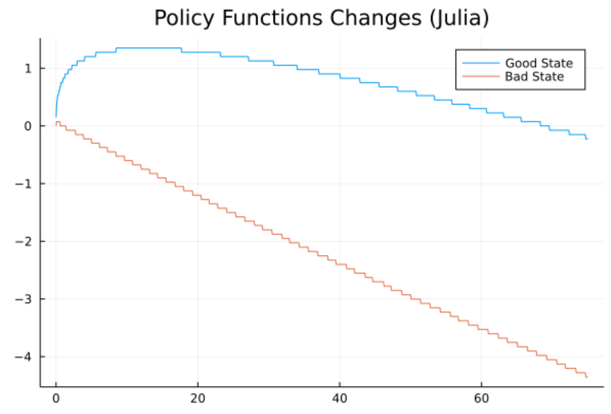

Figure 5: Policy Func. Changes from Julia

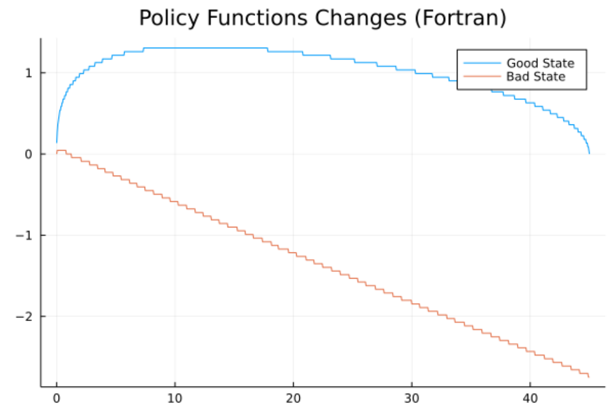

Figure 6: Policy Func. Changes from Fortran

Comparison of Computation Time: In addition to visualizing the results, we also compare the efficiency of different codes (julia, fortran, and julia parallelized). I ran all codes on my laptop, which has 4 cores. The table below summarizes the computation times:

Julia	Julia (Parallelized)	Fortran
6.443 seconds	6.355 seconds	13.648 seconds

As shown above, the parallelized version of julia solves the dynamic programming program the quickest out of the three. All julia and fortran codes are attached with this assignment.